Serialization

Sending Complex Java Data Structures to Files or Over the Network

Originals of slides and source code for examples: http://courses.coreservlets.com/Course-Materials/java.html
Also see Java 8 tutorial: http://www.coreservlets.com/java-8-tutorial/ and many other Java EE tutorials: http://www.coreservlets.com/
Customized Java training courses (onsite or at public venues): http://courses.coreservlets.com/java-training.html

For additional materials, please see http://www.coreservlets.com/. The Java tutorial section contains complete source code for all examples in this tutorial series, plus exercises and exercise solutions for each topic.

coreservlets.com – custom onsite training

For customized training related to Java or JavaScript, please email hall@coreservlets.com

Marty is also available for consulting and development support

The instructor is author of several popular Java EE books, two of the most popular Safari videos on Java and JavaScript, and this tutorial.

Courses available at public venues, or custom versions can be held on-site at your organization.

- **Courses developed and taught by Marty Hall**
  - JSF 2.3, PrimeFaces, Java programming (using Java 8, for those new to Java), Java 8 (for Java 7 programmers), JavaScript, jQuery, Angular 2, Ext JS, Spring Framework, Spring MVC, Android, GWT, custom mix of topics.
  - Java 9 training coming soon.
  - Courses available in any state or country.
  - Maryland/DC companies can also choose afternoon/evening courses.
- **Courses developed and taught by coreservlets.com experts (edited by Marty)**
  - Hadoop, Spark, Hibernate/JPA, HTML5, RESTful Web Services

Contact hall@coreservlets.com for details

Topics in This Section

- **Idea**
- **Requirements**
- **Steps for sending data**
- **Steps for receiving data**
- **Example: saving GUI in file**
  - And reading it from file later
- **Example: sending GUI across network**
  - And recreating it at the other end







Overview

Idea of Serialization

- **Java lets you send arbitrarily complex data structures with a single command**
  - writeObject from ObjectOutputStream
    - Can write to file, socket, process, etc.
  - Almost any data type: ArrayList, array, Frame, Panel, custom classes, etc. Arbitrarily nested.
    - Custom classes must implement Serializable
- **Java lets you read complex data structures in a single command**
  - readObject from ObjectInputStream
    - Can read from file, socket, process, etc.
  - Receiver must have class files for your custom classes
  - Receiver must be on same major version of Java

Benefits

- **Serialization is easy to use**

```
Shape[] shapes = { new Circle(…), … };
myObjectStream.writeObject(shapes);
```

- **Serialization is efficient**
  - What does this code do?

```
String s1 = …;
String s2 = …;
String[] strings = { s1, s2, s1, s2 };
if (strings[0] == strings[2]) {
  doThis();
} else {
  doThat();
}
```

Requirements for Using Serialization

- **Top-level data structure and all internal components must implement Serializable**
  - But most builtin classes already implement it.
  - Bottom-most non-Serializable class must have a zero-argument constructor. (I.e., parent of first Serializable class. Object *does* have zero-arg constructor.)
- **Both ends must use same version of Java**
  - I.e., sender cannot use Java 7 and receiver use Java 8 or vice versa. Minor versions (e.g., JDK 1.8.0_45 vs. JDK 1.8.0_46) do not matter.
- **Both ends must have same class files**
  - E.g., if you add a method to your class, old serialized data is no longer valid. But see serialVersionUID slide.

The Serializable Interface

- **Making classes serializable is simple**
  - Just say "implements Serializable"
    - Serializable has no methods! It is just a compiler marker.
  - Serializable is in the java.io package
- **Most builtin classes already implement Serializable**
  - ArrayList, HashMap, array, String, Frame/JFrame, Panel/JPanel, Button/JButton, etc.
- **Primitives are OK inside data structures**
  - No need for wrapper classes
- **Mark fields that can be ignored with transient**
  - private transient Thread myThread = …;
  - private transient int myRandomNumber = ...;

The serialVersionUID Field

- **Purpose**
  - To determine if the sender and receiver's class files are compatible
- **If no serialVersionUID**
  - Java creates one by conservatively analyzing the class
  - Even minor changes to class can result in a different id
- **If serialVersionUID declared**
  - Java compares the ids to see if the class files are compatible. Update the id when you make changes that invalidate saved data
- **Example**
  - private static final long serialVersionUID = 1L;

Sending Data

Sending Data: Summary

- **Wrap an ObjectOutputStream around any regular OutputStream**
  - To file

```
FileOutputStream fileOut = new FileOutputStream("SomeFile.ser");
ObjectOutputStream out = new ObjectOutputStream(fileOut);
```

  - To socket

```
OutputStream socketOut = someSocket.getOutputStream();
ObjectOutputStream out = new ObjectOutputStream(socketOut);
```

- **Send top-level data structure**

```
out.writeObject(theData);
out.close();
```


Sending Data to File: Details (Example for Array of Shapes)

```
try(FileOutputStream fileOut = new FileOutputStream("shapes.ser");
    ObjectOutputStream out = new ObjectOutputStream(fileOut)) {
  Shape[] shapes = { new Circle(...), new Rectangle(...), ...};
  out.writeObject(shapes);
} catch(IOException ioe) {
  System.out.println("Error sending data" + ioe);
}
```

FileOutputStream implements AutoCloseable, so you can use try-with-resources (in Java 7 and later) and skip the explicit call to close.

Also note that FileOutputStream, FileInputStream, ObjectOutputStream, ObjectInputStream, and PrintStream have very little to do with the very cool and powerful Stream interface introduced in Java 8 and covered in separate tutorial sections.

Sending Data to Socket : Details (Example for Array of Shapes)

```
try(Socket socket = new Socket("host", port);
    // Or Socket socket = serverSock.accept();
    OutputStream socketOut = socket.getOutputStream();
    ObjectOutputStream out = new ObjectOutputStream(socketOut)) {
  Shape[] shapes = { new Circle(...), new Rectangle(...), ...};
  out.writeObject(shapes);
} catch(IOException ioe) {
  System.out.println("Error sending data" + ioe);
}
```







Receiving Data

Receiving Data: Summary

- **Wrap an ObjectInputStream around any regular InputStream**
  - From file

```
FileInputStream fileIn =
  new FileInputStream(new File("SomeFile.ser"));
ObjectInputStream in =
  new ObjectInputStream(fileIn);
```

  - From socket

```
InputStream socketIn =
  someSocket.getInputStream();
ObjectInputStream in =
  new ObjectInputStream(socketIn);
```

- **Read top-level data structure**

```
SomeType var = (SomeType)in.readObject();
```


Reading Data from File: Details (Example for Array of Shapes)

```
try(FileInputStream fileIn =
      new FileInputStream(new File("shapes.ser"));
    ObjectInputStream in = new ObjectInputStream(fileIn)) {
  Shape[] shapes = (Shape[])in.readObject();
} catch(IOException ioe) {
  System.out.println("Error reading file: " + ioe);
} catch(ClassNotFoundException cnfe) {
  System.out.println("No such class: " + cnfe);
}
```

FileInputStream implements AutoCloseable, so you can use try-with-resources (in Java 7 and later) and skip the explicit call to close.

Reading Data from Socket: Details (Example for Array of Shapes)

```
try(Socket socket = new Socket("host", port);
    // Or Socket socket = serverSock.accept();
    InputStream socketIn = socket.getInputStream();
    ObjectInputStream in = new ObjectInputStream(socketIn)) {
  Shape[] shapes = (Shape[])in.readObject();
} catch(IOException ioe) {
  System.out.println("Error reading socket: " + ioe);
} catch(ClassNotFoundException cnfe) {
  System.out.println("No such class: " + cnfe);
}
```


Example: Sending Entire Window to File or Network

Example: SaveableFrame

- **Data:**
  - Main Frame (Frame already Serializable)
  - Frame has internal fields (ints) representing width, height, colors, layout manager, and location on screen
  - Two subpanels (Panel already Serializable)
  - Bottom panel has 2 buttons (Button already Serializable)
  - Top panel has:
    - Custom mouse listener that explicitly implements Serializable
    - BetterCircle objects that are created when user presses mouse. (Extends Component, which already implements Serializable)
- **Sending to/from file**
  - Clicking "Save" sends state of Frame to file.
  - If file exists when program starts, old state taken from file
- **Sending to/from network**
  - Server created that sends state of Frame to any client
  - Client created that connects to server and gets copy of Frame

SaveableFrame (Custom Class)

```
import java.awt.*;
import java.awt.event.*;
import java.io.*;

public class CirclePanel extends Panel {
  private class ClickAdapter extends MouseAdapter implements Serializable {
    public void mousePressed(MouseEvent event) {
      BetterCircle circle = new BetterCircle(Color.BLACK, 25);
      add(circle);
      circle.setCenter(event.getX(), event.getY());
    }
  }

  public CirclePanel() {
    setLayout(null);
    addMouseListener(new ClickAdapter());
  }
}
```

Already Serializable

Not already Serializable

SaveableFrame (Base Code to Send Frame)

- **SaveableFrame.java**

```
public void sendFrame(OutputStream rawOut) {
  try(ObjectOutputStream out = new ObjectOutputStream(rawOut)) {
    out.writeObject(this);
  } catch(IOException ioe) {
    System.out.println("Error saving frame: " + ioe);
  }
}
```


SaveableFrame (Code to Send Frame to File)

- **SaveableFrame.java**

```
@Override
public void actionPerformed(ActionEvent event) {
  if (event.getSource() == clearButton) {
    circlePanel.removeAll();
    circlePanel.repaint();
  } else if (event.getSource() == saveButton) {
    try(FileOutputStream fileOut =
          new FileOutputStream("SavedFrame.ser")) {
      sendFrame(fileOut);
    } catch(IOException ioe) {
      System.out.println("IOException: " + ioe);
    }
  }
}
```

SaveableFrame (Code to Send Frame to Client on Network)

- **FrameServer.java**

```
public void listen(int port, SaveableFrame frame) {
  try(ServerSocket listener = new ServerSocket(port)) {
    Socket server;
    while(true) {
      server = listener.accept();
      frame.sendFrame(server.getOutputStream());
      server.close();
    }
  } catch (IOException ioe) {
    System.out.println("IOException: " + ioe);
    ioe.printStackTrace();
  }
}
```


SaveableFrame (Base Code to Get Frame)

- **SaveableFrame.java**

```
public static SaveableFrame getFrame(InputStream rawIn) {
  SaveableFrame frame = null;
  try(ObjectInputStream in = new ObjectInputStream(rawIn)) {
    frame = (SaveableFrame)in.readObject();
    frame.setVisible(true);
    return(frame);
  } catch(IOException ioe) {
    System.out.println("Error reading file: " + ioe);
  } catch(ClassNotFoundException cnfe) {
    System.out.println("No such class: " + cnfe);
  }
  return(frame);
}
```

SaveableFrame (Code to Get Frame from File)

- **SaveableFrame.java**

```
public static void main(String[] args) {
  SaveableFrame frame;
  File serializeFile = new File(serializeFilename);
  if (serializeFile.exists()) {
    try(FileInputStream fileIn =
          new FileInputStream(serializeFile)) {
      frame = getFrame(fileIn);
    } catch(IOException ioe) {
      System.out.println("IOException: " + ioe);
    }
  } else {
    frame = new SaveableFrame();
  }
}
```

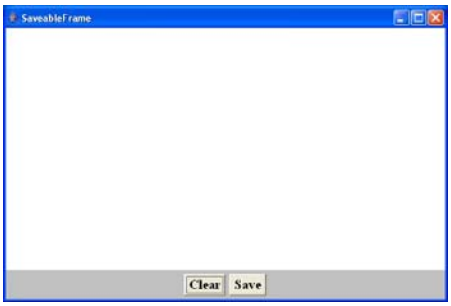

SaveableFrame (Code to Get Frame from Server on Network)

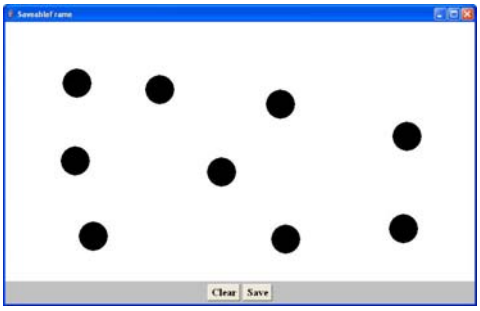
- **FrameClient.java**

```
public FrameClient(String host, int port) {
  try(Socket client = new Socket(host, port)) {
    SaveableFrame frame =
      SaveableFrame.getFrame(client.getInputStream());
  } catch(UnknownHostException uhe) {
    System.out.println("Unknown host: " + host);
    uhe.printStackTrace();
  } catch(IOException ioe) {
    System.out.println("IOException: " + ioe);
    ioe.printStackTrace();
  }
}
```

Results: SaveableFrame (Serialization to/from File)

- **Saving to File**
  - Open frame (600x400, no circles, top left corner)
  - Move window around
  - Resize it
  - Click to add circles
  - Press "Save"
- **Next time program runs**
  - Frame pops up at previous location, with previous size, including previous circles





Results: SaveableFrame (Serialization to/from Network)

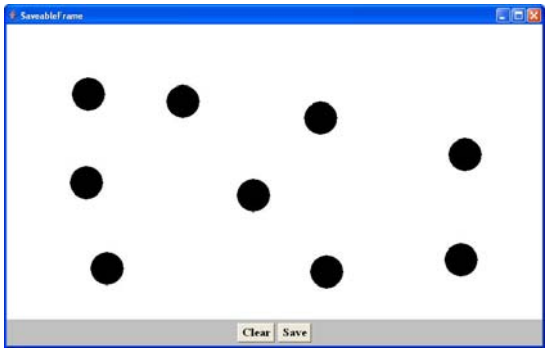
- **Machine 1**

```
> java FrameServer 8888
```

  - Open frame (600x400, no circles, top left corner)
  - Move window around
  - Resize it
  - Click to add circles
- **Machine 2**

```
> java FrameClient coreservlets.com 8888
```

  - Frame pops up with same location, size, and circles as version on the server



Wrap-Up

Summary

- **Class format**
  - Make sure custom classes implement Serializable and parent (non-Serializable) class has zero-arg constructor
    - Object has zero-arg constructor
- **Sending data**
  - Wrap an ObjectOutputStream around a raw OutputStream
  - Call writeObject(objectYouWantToSend)
    - You need to use try/catch blocks
- **Receiving data**
  - Wrap an ObjectInputStream around a raw InputStream
  - Call readObject
  - Cast the result to desired type
    - You need to use try/catch blocks

coreservlets.com – custom onsite training





Questions?

More info:
http://courses.coreservlets.com/Course-Materials/java.html – General Java programming tutorial
http://www.coreservlets.com/java-8-tutorial/ – Java 8 tutorial
http://courses.coreservlets.com/java-training.html – Customized Java training courses, at public venues or onsite at *your* organization
http://coreservlets.com/ – JSF 2, PrimeFaces, Java 7 or 8, Ajax, jQuery, Hadoop, RESTful Web Services, Android, HTML5, Spring, Hibernate, Servlets, JSP, GWT, and other Java EE training
Many additional free tutorials at coreservlets.com (JSF, Android, Ajax, Hadoop, and lots more)

For additional materials, please see http://www.coreservlets.com/. The Java tutorial section contains complete source code for all examples in this tutorial series, plus exercises and exercise solutions for each topic.