tick2 submission from David Brazdil

Name	David Brazdil (db538)
College	TRINH
Submission contents	uk/ac/cam/cl/fjava/messages/ChangeNickMessage.java uk/ac/cam/cl/fjava/messages/StatusMessage.java uk/ac/cam/cl/fjava/messages/RelayMessage.java uk/ac/cam/cl/fjava/messages/ChatMessage.java uk/ac/cam/cl/fjava/messages/Message.java uk/ac/cam/cl/fjava/messages/Execute.java uk/ac/cam/cl/fjava/messages/NewMessageType.java uk/ac/cam/cl/fjava/messages/DynamicObjectInputStream.java uk/ac/cam/db538/fjava/tick2/ChatClient.java uk/ac/cam/db538/fjava/tick2/FurtherJavaPreamble.java uk/ac/cam/db538/fjava/tick2/TestMessage.java uk/ac/cam/db538/fjava/tick2/TestMessageReadWrite.java
Ticker	Not yet assigned
Ticker signature	

ChangeNickMessage.java

```
package uk.ac.cam.cl.fjava.messages;

import java.io.Serializable;

public class ChangeNickMessage extends Message implements Serializable {
private static final long serialVersionUID = 1L;

public String name;

public ChangeNickMessage(String name) {
super();
this.name = name;
}

}
```

StatusMessage.java

```
package uk.ac.cam.cl.fjava.messages;

import java.io.Serializable;

public class StatusMessage extends Message implements Serializable {

private static final long serialVersionUID = 1L;
private String message;

public StatusMessage(String message) {
super();
this.message = message;
}

public String getMessage() {
return message;
}

}
```

RelayMessage.java

```java
package uk.ac.cam.cl.fjava.messages;

import java.io.Serializable;
import java.util.Date;

public class RelayMessage extends Message implements Serializable {

private static final long serialVersionUID = 1L;
private String from;
private String message;

public RelayMessage(String from, ChatMessage original) {
super(original);
this.from = from;
this.message = original.getMessage();
}

public RelayMessage(String from, String message, Date time) {
super(time);
this.from = from;
this.message = message;
}

public String getFrom() {
return from;
}

public String getMessage() {
return message;
}
}
```

ChatMessage.java

```java
package uk.ac.cam.cl.fjava.messages;

import java.io.Serializable;

/**
 * Message sent from the client to the server
 *
 */
public class ChatMessage extends Message implements Serializable {

private static final long serialVersionUID = 1L;
private String message;

public ChatMessage(String message) {
super();
this.message = message;
}

public String getMessage() {
return message;
}
}
```

Message.java

```
package uk.ac.cam.cl.fjava.messages;

import java.io.Serializable;
import java.util.Date;

public class Message implements Serializable {

private static final long serialVersionUID = 1L;
private Date creationTime;

public Message() {
creationTime = new Date();
}

protected Message(Message copy) {
creationTime = copy.creationTime;
}

protected Message(Date time) {
creationTime = time;
}

public Date getCreationTime() {
return creationTime;
}
}
```

Execute.java

```
package uk.ac.cam.cl.fjava.messages;

import java.lang.annotation.Retention;
import java.lang.annotation.RetentionPolicy;

//This is an "annotation". This is explained later Workbook 2
@Retention(RetentionPolicy.RUNTIME)
public @interface Execute {}
```

NewMessageType.java

```
package uk.ac.cam.cl.fjava.messages;


public class NewMessageType extends Message {

private static final long serialVersionUID = 1L;
private String name;
private byte[] classData;

public NewMessageType(String name, byte[] classData) {
super();
this.name = name;
this.classData = classData;
}

public String getName() {
return name;
}

public byte[] getClassData() {
return classData;
}

}
```

DynamicObjectInputStream.java

```
package uk.ac.cam.cl.fjava.messages;

import java.io.IOException;
import java.io.InputStream;
import java.io.ObjectInputStream;
import java.io.ObjectStreamClass;

public class DynamicObjectInputStream extends ObjectInputStream {

private ClassLoader current = ClassLoader.getSystemClassLoader();

public DynamicObjectInputStream(InputStream in) throws IOException {
super(in);
}

@Override
protected Class<?> resolveClass(ObjectStreamClass desc) throws IOException,
ClassNotFoundException {
try {
return current.loadClass(desc.getName());
}
catch (ClassNotFoundException e) {
return super.resolveClass(desc);
}
}

public void addClass(final String name, final byte[] defn) {
current = new ClassLoader(current) {
@Override
protected Class<?> findClass(String className)
throws ClassNotFoundException {
if (className.equals(name)) {
Class<?> result = defineClass(name, defn, 0, defn.length);
return result;
} else {
throw new ClassNotFoundException();
}
}
};
}

}
```

ChatClient.java

```
package uk.ac.cam.db538.fjava.tick2;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.ObjectOutputStream;
import java.lang.reflect.Field;
import java.lang.reflect.InvocationTargetException;
import java.lang.reflect.Method;
import java.net.Socket;
import java.net.UnknownHostException;
import java.text.SimpleDateFormat;
import java.util.Date;

import uk.ac.cam.cl.fjava.messages.ChangeNickMessage;
import uk.ac.cam.cl.fjava.messages.ChatMessage;
import uk.ac.cam.cl.fjava.messages.DynamicObjectInputStream;
import uk.ac.cam.cl.fjava.messages.Execute;
import uk.ac.cam.cl.fjava.messages.NewMessageType;
import uk.ac.cam.cl.fjava.messages.RelayMessage;
import uk.ac.cam.cl.fjava.messages.StatusMessage;
import uk.ac.cam.db538.fjava.tick2.FurtherJavaPreamble.Ticker;

@FurtherJavaPreamble(author = "David Brazdil",
                     crsid = "db538",
                     date = "07/11/2011",
                     summary = "ChatClient from Workbook 2",
                     ticker = Ticker.A)
public class ChatClient {
static void print(Date when, String from, String what) {
System.out.println(
new SimpleDateFormat("HH:mm:ss").format(when) +
" [" + from + "] " + what);
}

public static void main(String[] args) {
String server = null;
int port = 0;

if (args.length != 2) {
System.err.println("This application requires two arguments: <machine> <port>");
return;
}

server = args[0];
try {
port = Integer.parseInt(args[1]);
} catch (NumberFormatException ex) {
System.err.println("This application requires two arguments: <machine> <port>");
return;
}

try {
final Socket s = new Socket(server, port);
print(new Date(), "Client", "Connected to " + server + " on port " + port + ".");

Thread output = new Thread() {
@Override
public void run() {
DynamicObjectInputStream in;
try {
in = new DynamicObjectInputStream(s.getInputStream());
} catch (IOException ex) {
print(new Date(), "Client", ex.getClass().getName() + ": " + ex.getMessage());
return;
}
while(!s.isClosed()) {
try {
Object result = in.readObject();
if (result instanceof StatusMessage) {
StatusMessage msg = (StatusMessage) result;
print(msg.getCreationTime(), "Server", msg.getMessage());
} else if (result instanceof RelayMessage) {
RelayMessage msg = (RelayMessage) result;
print(msg.getCreationTime(), msg.getFrom(), msg.getMessage());
} else if (result instanceof NewMessageType) {
NewMessageType msg = (NewMessageType) result;
in.addClass(msg.getName(), msg.getClassData());
print(msg.getCreationTime(), "Client", "New class " + msg.getName() + " loaded.");
} else {
String text = result.getClass().getSimpleName() + ": ";
Field[] fields = result.getClass().getDeclaredFields();
for (int i = 0; i < fields.length; ++i) {
try {
fields[i].setAccessible(true);
text += fields[i].getName() + "(" + fields[i].get(result).toString() + "), ";
} catch (IllegalArgumentException e) {
} catch (IllegalAccessException e) {
e.printStackTrace();
}
}
if (text.endsWith(", "))
text = text.substring(0, text.length() - 2);
print(new Date(), "Client", text);

Method[] methods = result.getClass().getDeclaredMethods();
for (int i = 0; i < methods.length; ++i)
if (methods[i].getParameterTypes().length == 0 &&
methods[i].isAnnotationPresent(Execute.class))
try {
methods[i].invoke(result);
} catch (IllegalArgumentException e) {
} catch (IllegalAccessException e) {
} catch (InvocationTargetException e) {
}
}
} catch (IOException ex) {
// print(new Date(), "Client", ex.getClass().getName() + ": " + ex.getMessage());
} catch (ClassNotFoundException e) {
print(new Date(), "Client", "New message of unknown type received.");
}
}
}
};
output.setDaemon(true);
output.start();

ObjectOutputStream out = new ObjectOutputStream(s.getOutputStream());
BufferedReader r = new BufferedReader(new InputStreamReader(System.in));
boolean go = true;
while (go) {
String line = r.readLine();
if (line.startsWith("\\")) {
// command mode
String[] lineSplit = line.split(" ");
String command = lineSplit[0].substring(1);
if (command.equals("quit"))
go = false;
else if (command.equals("nick")) {
if (lineSplit.length != 2)
print(new Date(), "Client", "The nick command requires one argument - new nickname");
else
out.writeObject(new ChangeNickMessage(lineSplit[1]));
} else
print(new Date(), "Client", "Unknwown command \"" + command + "\"");
} else
out.writeObject(new ChatMessage(line));
}

s.close();
print(new Date(), "Client", "Connection terminated.");
} catch (NumberFormatException e) {
System.err.println("Cannot connect to " + server + " on port " + port);
} catch (UnknownHostException e) {
System.err.println("Cannot connect to " + server + " on port " + port);
```

```
} catch (IOException e) {
System.err.println("Cannot connect to " + server + " on port " + port);
}
}
}
```

FurtherJavaPreamble.java

```
package uk.ac.cam.db538.fjava.tick2;

import java.lang.annotation.Retention;
import java.lang.annotation.RetentionPolicy;

@Retention(RetentionPolicy.RUNTIME)
public @interface FurtherJavaPreamble {
enum Ticker {A, B, C, D};
String author();
String date();
String crsid();
String summary();
Ticker ticker();
}
```

TestMessage.java

```
package uk.ac.cam.db538.fjava.tick2;

import java.io.Serializable;

public class TestMessage implements Serializable {
private static final long serialVersionUID = 1L;

private String text;
public String getMessage() { return text; }
public void setMessage(String msg) { this.text = msg; }
}
```

TestMessageReadWrite.java

```java
package uk.ac.cam.db538.fjava.tick2;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.net.MalformedURLException;
import java.net.URL;

public class TestMessageReadWrite {
static boolean writeMessage(String message, String filename) {
TestMessage msgObject = new TestMessage();
msgObject.setMessage(message);

try {
FileOutputStream fos = new FileOutputStream(filename);
ObjectOutputStream out = new ObjectOutputStream(fos);
out.writeObject(msgObject);
out.close();
} catch (FileNotFoundException ex) {
return false;
} catch (IOException e) {
return false;
}

return true;
}

static String readMessage(String location) {
try {
InputStream stream = null;
if (location.startsWith("http://"))
stream = new URL(location).openConnection().getInputStream();
else
stream = new FileInputStream(new File(location));

ObjectInputStream in = new ObjectInputStream(stream);
Object result = in.readObject();
if (result instanceof TestMessage)
return ((TestMessage) result).getMessage();
return null;
} catch (FileNotFoundException ex) {
return null;
} catch (MalformedURLException e) {
return null;
} catch (IOException e) {
return null;
} catch (ClassNotFoundException e) {
return null;
}
}

public static void main(String[] args) {
System.out.println(readMessage(args[0]));
writeMessage("You are a piece of shit!!!", "test.jobj");
System.out.println(readMessage("test.jobj"));
}
}
```