CS 580 Client-Server Programming
Spring Semester, 2005
Assignment 4 Comments

Contents

System.exit(0);.................................... 2
getId() Verses isX().............................. 2
Comment = method............................. 3
Coupling Message to Socket.................... 4
Repeated Code in Message Subclasses...... 5

System.exit(0);

Avoid using System.exit();

The rest of the program may need to perform some task

System.out.println()

Logger anyone?

getId() Verses isX()

```
if (message.getId() == Message.SEARCH_RESULT_ID )
    blah
```

```
if (message.isSearchResult() )
    blah
```

Comment = method

```
// get the length as an integer value
int length = 0;
for (int k = 0; k < 4; k++ )
    blah
    blah

// Get the rest of the message
message = new byte[length];
while( blah )
    more blah
```

```
int length = readLength();
message = readBytes(length);
```

Coupling Message to Socket

Hard to test the following - it sends things to the server

```
public class Handshake {
    public Handshake(Socket toServer) {
        blah
    }
    blah
}
```

The following can be tested without server

Create a OutputStream that writes to a string/byte array

```
public class Handshake {
    public Handshake(OutputStream toServer) {
        blah
    }
    blah
}
```

This is easier to test

Can look at the output directly

```
public class Handshake {
    public byte[] toBytes() {
        blah
    }
}
```

Repeated Code in Message Subclasses

```
public class HandshakeMessage extends Message {

    public byte[] toBytes() throws SomeException {
        byte[] messageBytes = new byte[version.length() + 1 + 4];
        int length = version.length() + 1;

        messageBytes[0] = (byte) ((length & 0xff000000) >>> 24);
        messageBytes[1] = (byte) ((length & 0x00ff0000) >>> 16);
        messageBytes[2] = (byte) ((length & 0x0000ff00) >>> 8);
        messageBytes[3] = (byte) (length & 0x000000ff);

        messageBytes[4] = (byte) 1;

        versionBytes = version.getBytes();
        for (int k=5, j=0; j < length; k++, j++ )
            messageBytes[k] = versionBytes[j];

        return messageBytes;
    }
}
```

```
public class ErrorMessage extends Message {

    public byte[] toBytes() throws SomeException {
        byte[] messageBytes = new byte[errorText.length() + 1 + 4];
        int length = version.length() + 1;

        messageBytes[0] = (byte) ((length & 0xff000000) >>> 24);
        messageBytes[1] = (byte) ((length & 0x00ff0000) >>> 16);
        messageBytes[2] = (byte) ((length & 0x0000ff00) >>> 8);
        messageBytes[3] = (byte) (length & 0x000000ff);

        messageBytes[4] = (byte) 9;

        errorBytes = errorText.getBytes();
        for (int k=5, j=0; j < length; k++, j++ )
            messageBytes[k] = errorBytes[j];

        return messageBytes;
    }
}
```

The methods are nearly identical!

Use Parent methods

```
public class Message {
    public void addLength(byte[] message, int length ) {
        message[0] = (byte) ((length & 0xff000000) >>> 24);
        message[1] = (byte) ((length & 0x00ff0000) >>> 16);
        message[2] = (byte) ((length & 0x0000ff00) >>> 8);
        message[3] = (byte) (length & 0x000000ff);
    }

    public void addId(byte[] message, int id ) {
        message[4] = (byte) id;
    }

    public void addPayload(byte[] message, byte[] payload) {
        System.arraycopy(payload,0,message, 5, payload.length);
    }
}

public class HandshakeMessage extends Message {

    public byte[] toBytes() throws SomeException {
        byte[] messageBytes = new byte[version.length() + 1 + 4];
        addLength(messageBytes, version.length() + 1);
        addId(messageBytes, 1);
        addPayload(messageBytes, version.getBytes());
        return messageBytes;
    }
}
```

toBytes method is shorter and avoids repeating some code

But still is has a lot of repeated code

Use Template Method

```
public class Message {

public byte[] toBytes() throws SomeException {
        byte[] payload = payload();
        byte id = id();

        byte[] messageBytes = new byte[payload.length() + 1 + 4];

        addLength(messageBytes, payload.length() + 1);
        addId(messageBytes, id);
        addPayload(messageBytes, payload);

        return messageBytes;
    }
}

public class HandshakeMessage extends Message {

    public byte[] payload() {
        return version.getBytes();
    }

    public byte id() {   return (byte) 1;  }
}

public class ErrorMessage extends Message {

    public byte[] payload() {
        return errorText.getBytes();
    }

    public byte id() {   return (byte) 9; }
}
```

Subclasses just supply parts that differ to parent's method