CS 580 Client-Server Programming
Spring Semester, 2005
Assignment 1 Comments
Assignment Index

San Diego State University -- This page last updated 2/10/05

Formating 2
Line Wrap 3

Leave Space Between Methods 4

Code Comments 5
Duh Comments 6
What Needs to be Commented? 7

Names 8
Name Structure Conventions 8

Coding Issues 9
Static Methods 9
What to return on Error? 10
Don't Replace Local Variables with Fields 12
Using Overloaded Method Names 13
How to Determine the Type 14

Smalltalk Solution In Java Syntax 15

Sorting the Keys of a Hashtable 16

Formating

Very Bad See Embarrassing Ways to Lose Points

Indent blocks consistently

```
private String encode(Object unBencodedObject) throws
UnsupportedObjectException
{
if (unBencodedObject instanceof String){
return encode((String) unBencodedObject);
}
else if (unBencodedObject instanceof Integer){
return encode((Integer) unBencodedObject);
}
else if (unBencodedObject instanceof List){
return encode((List) unBencodedObject);
}
else if (unBencodedObject instanceof Map){
return encode((Map) unBencodedObject);
}
else{
throw new UnsupportedObjectException();
}
```

Line Wrap

Bad See Embarrassing Ways to Lose Points

```
public void setOfficeData(Vector officeData) {

   if (_facultyInformationView.getCancelButton().
isEnabled()) {
       if (officeData.size() == 3) {
       boolean clearTable = false;
       int rows = _facultyInformationView.
getOfficeDataTable().getRowCount();

       for (int i=0; i<rows; i++) {
         if ( ((String)_facultyInformationView.
getOfficeDataTable ().getValueAt( i, 0)).equals("")) {
           _facultyInformationView.getOfficeDataTable().
setValueAt( officeData.get(0), i, 0);
           _facultyInformationView.getOfficeDataTable().
setValueAt( officeData.get(1), i, 1);
           _facultyInformationView.getOfficeDataTable().
setValueAt( officeData.get(2), i, 2);
           break;
```

Leave Space Between Methods

```
public String encode(String unBencodedString) {
  return unBencodedString.length() + ":" + unBencodedString;
}
public String encode(Integer unBencodedInteger) {
  return "i" + unBencodedInteger.toString() + "e";
}
```

```
public String encode(String unBencodedString) {
  return unBencodedString.length() + ":" + unBencodedString;
}

public String encode(Integer unBencodedInteger) {
  return "i" + unBencodedInteger.toString() + "e";
}
```

Code Comments

“Comments are easier to write poorly that well, and commenting can be more damaging than helpful.”

“Too many comments are as bad as too few...”

“The main contributor to code-level documentation isn’t comments, but good programming style. Style include good program structure, use of straightforward and easily understandable approaches, good variable names, good routine names, use of named constants instead of literals, clear layout, ...”

Steve McConnell, Code Complete 2

Duh Comments

```
x = x + 1;  //Add one to x
```

```
public class  Bencoding {

  //Empty Constructor
  public Bencoding() {
  }

  /**
  * Function: String encode(String input)
  * Encodes a String in Bencode format
  * @param input String
  * @return String
  */
  public String encode(String input) {
```

Why just repeat the code?

What Needs to be Commented?

Names

Select names that indicate role of variables

Name Structure Conventions

Type	C#	Java	Smalltalk
Class	PascalCase	PascalCase	PascalCase
Constant	PascalCase	ALL_CAPS	NA
Method	PascalCase	camelCase	camelCase
Namespace	PascalCase	all lower case	PascalCase
Property	PascalCase	NA	NA
Instance Field	camelCase	camelCase	camelCase
Parameter	camelCase	camelCase	camelCase
Local variable	camelCase	camelCase	camelCase

Coding Issues
Static Methods

```
public class  Bencoding {

  public static String encode(String unBencodedString) {
    return unBencodedString.length() + ":" + unBencodedString;
  }

  public static String encode(Integer unBencodedInteger) {
    return "i" + unBencodedInteger.toString() + "e";
  }
```

Avoid static methods - restrict use of objects

What to return on Error?

```
public class Bencoding
{
  int stringLength;

  public String encode(String unBencodedString)
  {
    if (unBencodedString != null)
    {
      if (unBencodedString == “” ) //Empty string
      {
        return unBencodedString = “0:”; //bencoded empty string
      }
      else
      {
        stringLength = unBencodedString.length();
        return stringLength + ":" + unBencodedString;
      }
    else
      return “ERROR: NULL STRING”; //Error string returned
  }
```

How does your code know an error occurred?

Use Exceptions

```
public class Bencoding
{
  int stringLength;

  public String encode(String unBencodedString)
  {
     if (unBencodedString == null)
        throw NullDataException;
     if (unBencodedString == “” )
        return “0:”;
     stringLength = unBencodedString.length();
     return stringLength + ":" + unBencodedString;
  }
```

Don’t Replace Local Variables with Fields

```
public class Bencoding
{
  int stringLength;

  public String encode(String unBencodedString)
  {
    if (unBencodedString == null)
      throw NullDataException;
    stringLength = unBencodedString.length();
    return stringLength + ":" + unBencodedString;
  }
```

Makes understanding class much harder

```
  public String encode(String unBencodedString)
  {
    if (unBencodedString == null)
      throw NullDataException;
    int stringLength = unBencodedString.length();
    return stringLength + ":" + unBencodedString;
  }
```

Using Overloaded Method Names

```
public String encodeString(String unBencodedString) {
    return unBencodedString.length() + ":" + unBencodedString;
  }

  public String encodeInteger(Integer unBencodedInteger) {
    return "i" + unBencodedInteger.toString() + "e";
  }
```

```
public String encode(String unBencodedString) {
    return unBencodedString.length() + ":" + unBencodedString;
  }

  public String encode(Integer unBencodedInteger) {
    return "i" + unBencodedInteger.toString() + "e";
  }
```

How to Determine the Type

```
public String encode(Object unBencodedObject)
  throws UnsupportedObjectException
  {
  if (unBencodedObject instanceof String)
    return encode((String) unBencodedObject);
  if (unBencodedObject instanceof Integer)
    return encode((Integer) unBencodedObject);
  if (unBencodedObject instanceof List)
    return encode((List) unBencodedObject);
  if (unBencodedObject instanceof Map)
    return encode((Map) unBencodedObject);

  throw new UnsupportedObjectException();
  }
```

Smalltalk Solution In Java Syntax

Only showing methods added to classes

```
public class Object {
  public String bencode() { throw UnsupportedObjectException;}
}

public class String {
  public String bencode() { return this.length() + ":" + this;}
}

public class Integer {
  public String bencode() { "i" + this + "e";}
}

etc.
```

To encode an object use:

```
x.bencode();
```

Sorting the Keys of a Hashtable

```
Hashtable unsortedKeys = new Hashtable();
blah to add values
TreeMap sortedKeys = new TreeMap(unsortedKeys);
```