CS 535 Object-Oriented Programming & Design
Fall Semester, 2000
Doc 6 Some Code Comments
Contents

References

Student papers from past semesters

Naming Issues
Example 1

```
public BooleanQuestion(String message,
   String methodName1,
   String methodName2,
   Object objectName1,
   Object objectName2 )

public BooleanQuestion(String message,
   String trueCallbackMethod,
   Object trueReceiver,
   String falseCallbackMethod,
   Object falseReceiver)
```

Numbered variables indicate order not the role the variable plays

Example 2

```
public class p1 {
   blah
   }
```

Set of Functions

```
pubic class Question {
   public String stringQuestion( String message ) {
      System.out.println( "message" );
      System.out.println( "Your answer: " );
      String response = Console.readLine().trim();
      return response;
   }

   public boolean booleanQuestion( String message ) {
      try {
         System.out.println( "message" );
         System.out.println( "Your answer: " );
         booelan response = Console.readBoolean();
         Console.flushLine();
         return response;
      } catch (NumberFormatError badInput ) { //blah}
   }

   public int intQuestion( String message ) {
      try { //blah
      } catch (NumberFormatError badInput ) {//more blah }
   }
}
```

What is the abstraction? Where is the data?

Relevant Heuristics

3.3 Beware of classes that have many accessor methods defined in their public interfaces. Having many implies that related data and behavior are not being kept in one place

A number of people had classes with mainly accessor methods.

2.9 Keep related data and behavior in one place

Not following 2.9 often leads to violating 3.3

2.8 A class should capture one and only one key abstraction

In the object-oriented world an abstraction includes data and behavior

What would the Data look like for BooleanQuestion?

```
public class BooleanQuestion extends ASCIIComponent
  {
  private static String[] trueAnswers = { "true", "t", "yes", "y" };
  private static String[] falseAnswers = { "false", "f", "no", "n" };
  private String text;
  private CallBackAdapter trueCallBack;
  private CallBackAdapter falseCallBack;
```

Keep UI Code separate

```
public class foo {
   int data;
   int value;

   public setData() {
      value = sdsu.io.Console.readInt( "Type an int for data" );
   }

   public void setValue(int aValue) {
      value = aValue;
   }
```

Contrast the two methods. The setData method uses Console, which uses System.out and System.in. The method is now coupled to those classes. It has to be used with a text based user interface. It also assumes that the user always enters the value for data. What happens if the value is computed? The setValue method is not coupled to the Screen. Other code will obtain the value and pass it to setValue. So the value can be obtained by asking user (via text based or graphical based interface), from a file, computed from other data, etc.

Cut & Pasting of Code

There are two major problems of cutting and pasting of code:

- Difficulty in changing maintaining the code
  The same code is repeated in many locations. Making it hard to change the code. It has to be changed in all N locations. Where are all N locations?

- Missing an abstraction
  Cutting and pasting code often means you are missing a useful abstraction. Better to find the abstraction.

Types of Things Repeated in Code

- Lines of Code
- Logic
- Algorithms

Repeating Lines of Code
Before

```
public void method1() {
   int foo = 12;
   //blah
   statement1;
   statement2;
   statement3;
   // more blah
}

public void method2() {
   int bar = 42;
   //stuff
   statement1;
   statement2;
   statement3;
   // more stuff
}
```

Repeating Lines of Code
After

Factoring out the common states sometimes is as easy as given here. Often it requires more thought. The common code operates on a set of values. How to get the values to and from the common method? Sometimes it takes a shift of point of view & abstraction to do this.

```
public void method1() {
   int foo = 12;
   //blah
   foo = repeated( whatGoesHere );
   // more blah
}

public void method2() {
   int bar = 42;
   //stuff
   bar = repeated( something );
   // more stuff
}

public int repeated(Object agrument) {
   statement1;
   statement2;
   statement3;
   return result;
}
```