Sample

public class StringParser...

public Node find(...) {

...

return StringNode.createStringNode(

textBuffer, textBegin, textEnd,

parser.shouldDecodeNodes()

);

}

public class StringNode...

public static Node createStringNode(

StringBuffer textBuffer, int textBegin, int textEnd, boolean shouldDecode) {

if (shouldDecode)

return new DecodingStringNode(

new StringNode(textBuffer, textBegin, textEnd)

);

return new StringNode(textBuffer, textBegin, textEnd);

}

Title: Refactoring: **Move method** in order to reduce or eliminate the dependency of the class calling the method on the class in which it’s located

Motivation to refactor:

When the knowledge for creating an object is spread out across numerous classes, you have **creation sprawl**: the placement of creational responsibilities in classes that ought not to be playing any role in an object’s creation. Creation sprawl, which is a case of the Solution Sprawl smell (43), tends to result from an earlier design problem

Pattern: The **Factory pattern** is helpful in this context. It uses one class to encapsulate both creation logic and a client’s instantiation/configuration preferences