ESOF 322 Homework 5

Kyle Rust

1. Question 1
    a. Part A
        i. I downloaded the open source project jEdit (see link to download it in the references section).
        ii. jEdit is a programmer's text editor written in Java. It uses the Swing toolkit for the GUI and can be configured to be an IDE using its plugin architecture.
        iii. The entire project is 196,479 lines of Java code specifically. I ran the following command in the WSL terminal to get these results: "find . -name '*.java'|xargs wc -l" This command went through the project folder and counted all the lines in each file that was of type Java.
    b. Part B
        i.

```
Found 47 files that possibly contain design patterns.

\org\gjt\sp\jedit\BeanShellFacade.java
Possible patterns: Facade

\org\gjt\sp\jedit\browser\BrowserCommandsMenu.java
Possible patterns: Command

\org\gjt\sp\jedit\bsh\BSHEnhancedForStatement.java
Possible patterns: State

\org\gjt\sp\jedit\bsh\BSHForStatement.java
Possible patterns: State

\org\gjt\sp\jedit\bsh\BSHIfStatement.java
Possible patterns: State

\org\gjt\sp\jedit\bsh\BshIterator.java
Possible patterns: Iterator

\org\gjt\sp\jedit\bsh\BSHReturnStatement.java
Possible patterns: State

\org\gjt\sp\jedit\bsh\BSHStatementExpressionList.java
Possible patterns: State

\org\gjt\sp\jedit\bsh\BSHSwitchStatement.java
Possible patterns: State

\org\gjt\sp\jedit\bsh\BSHThrowStatement.java
Possible patterns: State

\org\gjt\sp\jedit\bsh\BSHTryStatement.java
Possible patterns: State

\org\gjt\sp\jedit\bsh\BSHWhileStatement.java
Possible patterns: State

\org\gjt\sp\jedit\bsh\CommandLineReader.java
Possible patterns: Command

\org\gjt\sp\jedit\bsh\Interpreter.java
Possible patterns: Interpreter

\org\gjt\sp\jedit\bsh\InterpreterError.java
Possible patterns: Interpreter

\org\gjt\sp\jedit\bsh\JJTParserState.java
Possible patterns: State

\org\gjt\sp\jedit\bsh\collection\CollectionIterator.java
Possible patterns: Iterator

\org\gjt\sp\jedit\bsh\org\objectweb\asm\ClassVisitor.java
Possible patterns: Visitor

\org\gjt\sp\jedit\bsh\org\objectweb\asm\CodeVisitor.java
Possible patterns: Visitor

\org\gjt\sp\jedit\buffer\BufferAdapter.java
Possible patterns: Adapter

\org\gjt\sp\jedit\bufferset\BufferSetAdapter.java
Possible patterns: Adapter

\org\gjt\sp\jedit\gui\DockableWindowFactory.java
Possible patterns: Factory

\org\gjt\sp\jedit\gui\statusbar\BufferSetWidgetFactory.java
Possible patterns: Factory

\org\gjt\sp\jedit\gui\statusbar\ClockWidgetFactory.java
Possible patterns: Factory

\org\gjt\sp\jedit\gui\statusbar\EncodingWidgetFactory.java
Possible patterns: Factory

\org\gjt\sp\jedit\gui\statusbar\ErrorsWidgetFactory.java
Possible patterns: Factory

\org\gjt\sp\jedit\gui\statusbar\FoldWidgetFactory.java
Possible patterns: Factory

\org\gjt\sp\jedit\gui\statusbar\IndentWidgetFactory.java
Possible patterns: Factory

\org\gjt\sp\jedit\gui\statusbar\LastModifiedWidgetFactory.java
Possible patterns: Factory

\org\gjt\sp\jedit\gui\statusbar\LineSepWidgetFactory.java
Possible patterns: Factory

\org\gjt\sp\jedit\gui\statusbar\LockedWidgetFactory.java
Possible patterns: Factory

\org\gjt\sp\jedit\gui\statusbar\MemoryStatusWidgetFactory.java
Possible patterns: Factory

\org\gjt\sp\jedit\gui\statusbar\ModeWidgetFactory.java
Possible patterns: Factory

\org\gjt\sp\jedit\gui\statusbar\MultiSelectWidgetFactory.java
Possible patterns: Factory

\org\gjt\sp\jedit\gui\statusbar\OverwriteWidgetFactory.java
Possible patterns: Factory

\org\gjt\sp\jedit\gui\statusbar\RectSelectWidgetFactory.java
Possible patterns: Factory

\org\gjt\sp\jedit\gui\statusbar\SelectionLengthWidgetFactory.java
Possible patterns: Factory

\org\gjt\sp\jedit\gui\statusbar\StatusWidgetFactory.java
Possible patterns: Factory

\org\gjt\sp\jedit\gui\statusbar\TaskMonitorWidgetFactory.java
Possible patterns: Factory

\org\gjt\sp\jedit\gui\statusbar\WrapWidgetFactory.java
Possible patterns: Factory

\org\gjt\sp\jedit\indent\IndentRuleFactory.java
Possible patterns: Factory

\org\gjt\sp\jedit\pluginmgr\KeyboardCommand.java
Possible patterns: Command

\org\gjt\sp\jedit\visitors\JEditVisitor.java
Possible patterns: Visitor

\org\gjt\sp\jedit\visitors\JEditVisitorAdapter.java
Possible patterns: Adapter, Visitor

\org\gjt\sp\jedit\visitors\SaveCaretInfoVisitor.java
Possible patterns: Visitor

\org\gjt\sp\util\ProgressObserver.java
Possible patterns: Observer

\org\gjt\sp\util\TaskAdapter.java
Possible patterns: Adapter
```

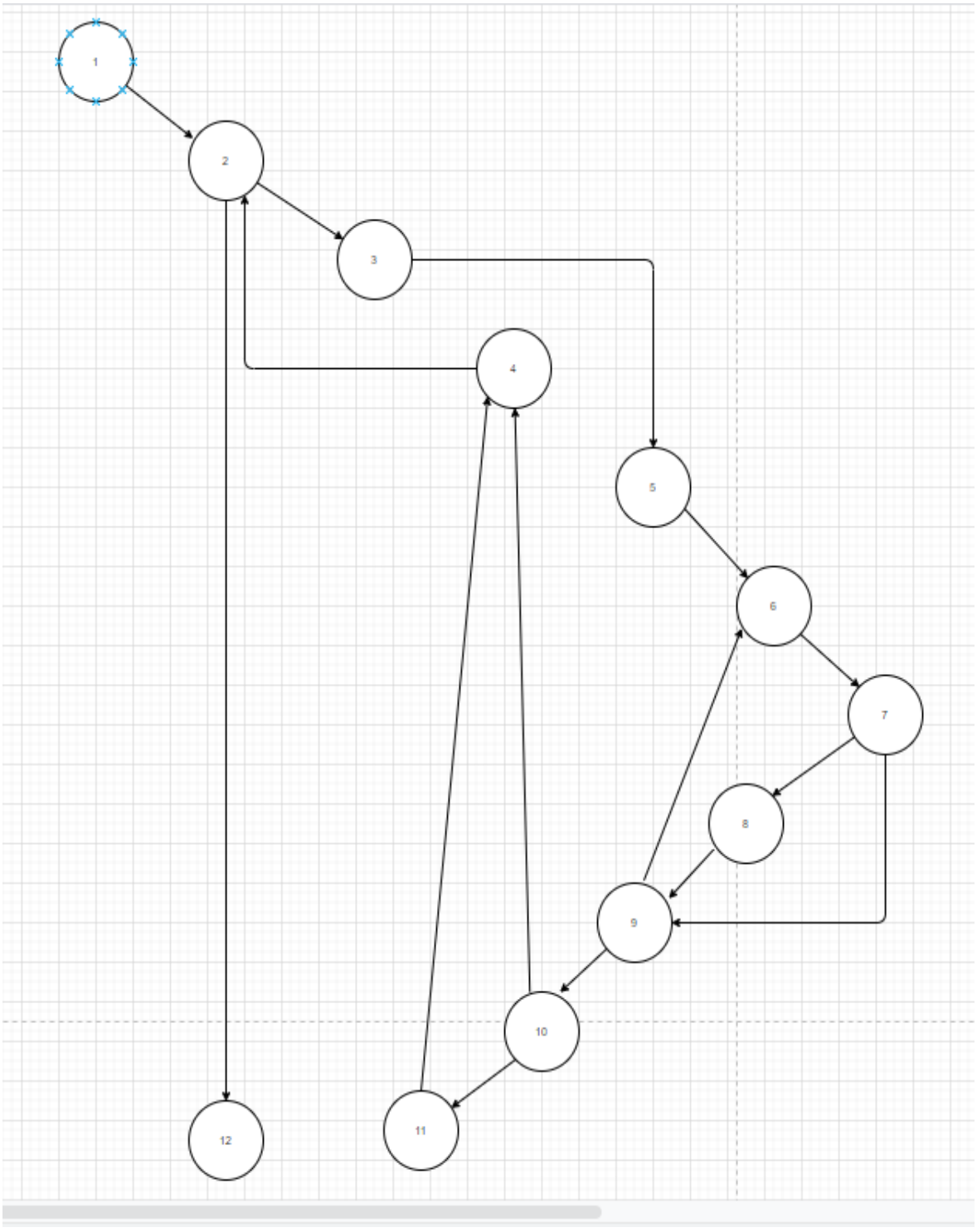
        ii. I think software uses a few key factors to identify if a design pattern is being used. The first thing I think it looks for is descriptive words in the file name. In every file that my open source project has, it has a pattern name in the file name. I think

the program looks at each file and first checks if a pattern is explicitly stated. This would cut down a lot of time needed to look at individual file structures if it came prelabeled. If the file is not clearly labeled, then I think the software looks for key features in a checklist. One the executable file is extracted, there is an XML file called Pattern Definitions. I believe the patterns are defined by their structure and relationships between classes. I think the files are checked for how the classes are connected to each other (inheritance, extends, etc.) and once a criterion for a pattern is located some sort of flag is set. If all of the flags in a given pattern are set, then the design pattern label is applied.

iii. Yes, I do believe the software pattern identifier. By examining my results, the pattern identified seems to match the pattern each file name would indicate. My research interest here at Montana State is machine learning and I see an opportunity for it to be applied in this situation. I would use a supervised approach to machine learning to accomplish this. How I would do it is to create a training dataset with patterns prelabeled. This means I would take a large amount of java files and assign them labels based on if they were apart of a design pattern or not. If they were included in a design pattern, I would include what pattern they were in. From there I would train a classification algorithm to identify features of the design patterns. Without actually completing this I am not sure exactly what algorithm I would use, perhaps a Random Forrest Classifier. Then I could deploy my trained model on data outside of my training data to test the effectiveness of my model.

2. Question 2

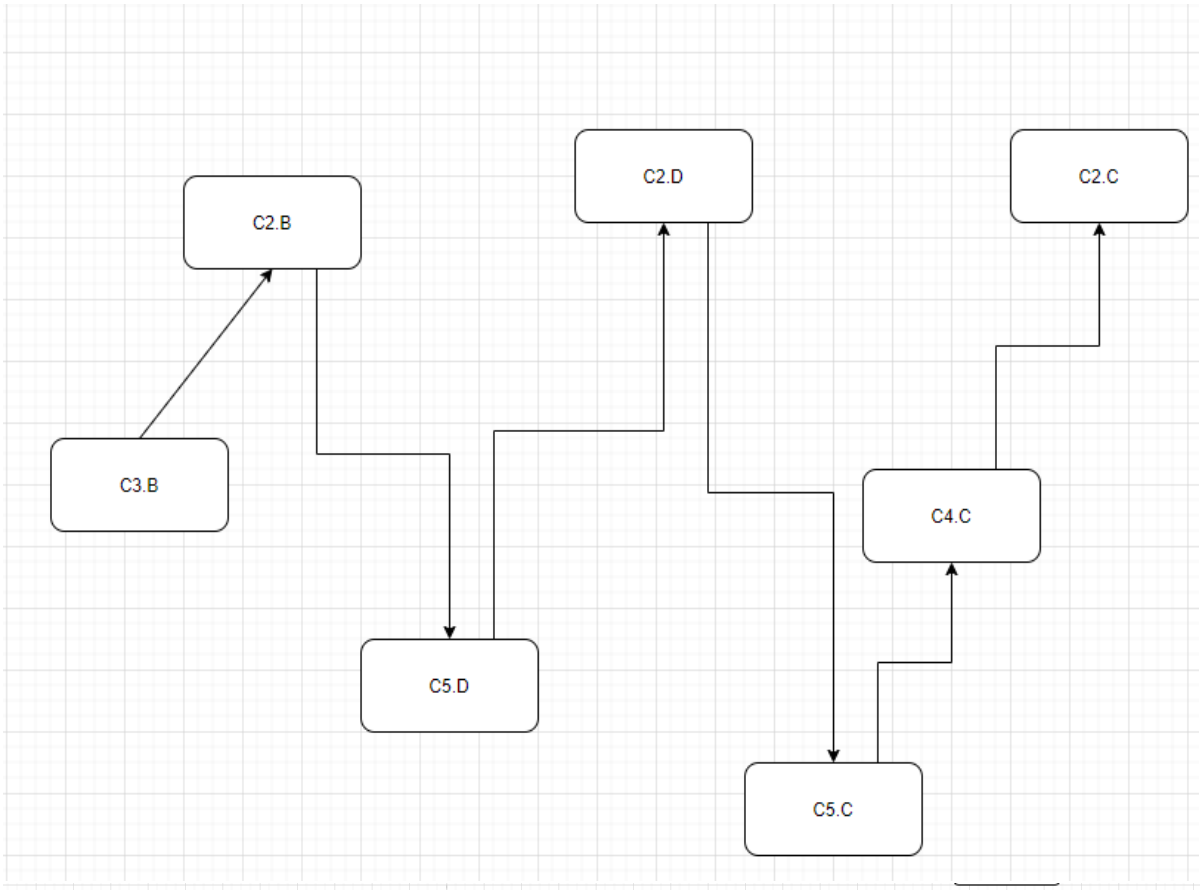
```
1. /* Find all primes from 2-upper_bound using Sieve of Eratosthanes */
2.
3. #include
4. typedef struct IntList {
5.           int value;
6.           struct IntList *next;
7.           } *INTLIST, INTCELL;
8. INTLIST sieve ( int upper_bound ) {
9.
10.        INTLIST prime_list = NULL;   /* list of primes found */
11.        INTLIST cursor;              /* cursor into prime list */
12.        int candidate;               /* a candidate prime number */
13.        int is_prime;                /* flag: 1=prime, 0=not prime */
14.
15.         /* try all numbers up to upper_bound */
16.         for (candidate=2;
17.
18.              candidate <= upper_bound;
19.              candidate++) {
20.
21.          is_prime = 1; /* assume candidate is prime */
22.          for(cursor = prime_list;
23.
24.               cursor;
25.               cursor = cursor->next) {
26.
27.             if (candidate % cursor->value == 0) {
28.
29.               /* candidate divisible by prime */
30.               /* in list, can't be prime */
31.               is_prime = 0;
32.               break;  /* "for cursor" loop */
33.             }
34.          }
35.          if(is_prime) {
36.
37.            /* add candidate to front of list */
38.            cursor = (INTLIST) malloc(sizeof(INTCELL));
39.            cursor->value = candidate;
40.            cursor->next  = prime_list;
41.            prime_list = cursor;
42.          }
43.         }
44.         return prime_list;
45.        }
```

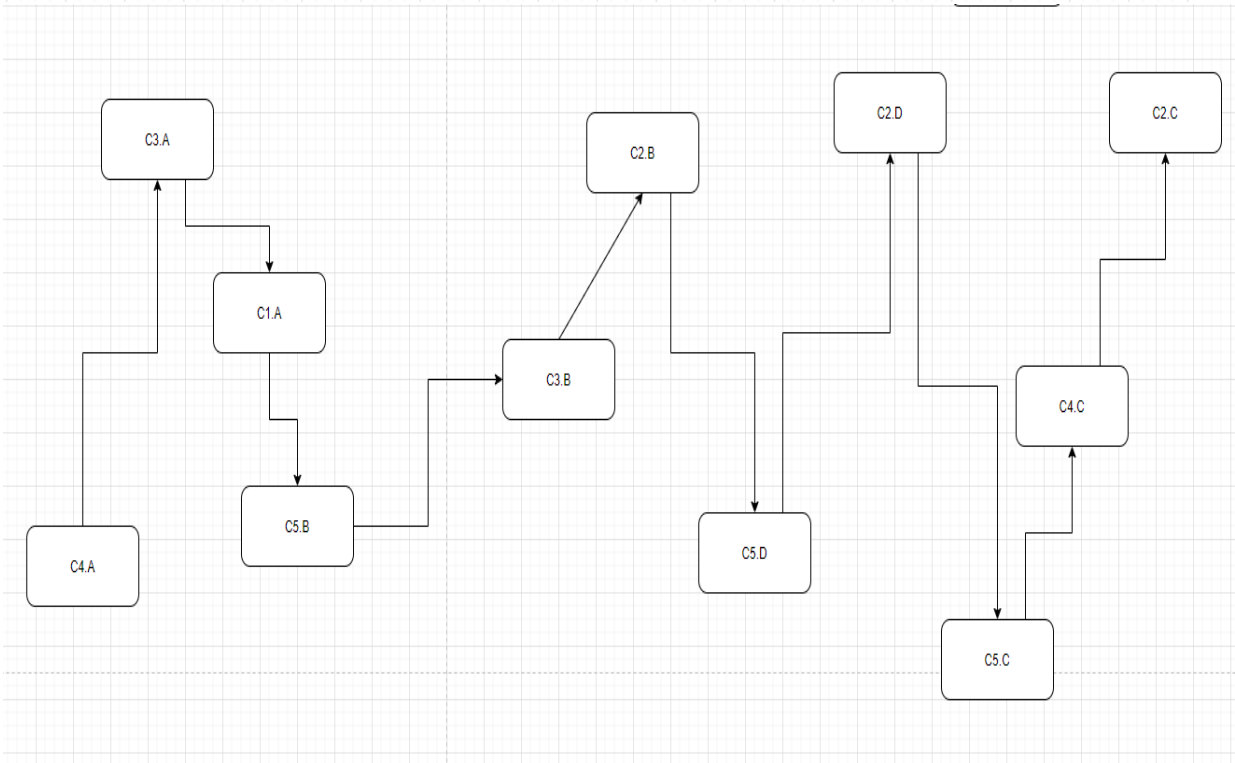


a. {<1,2,3,5,6,7,8,9,10,11,4,12>}
b. {<1,2,3,5,6,7,9,10,4,2,12> <1,2,3,5,6,7,8,9,6,7,9,10,11,4,2,12>}
c. 100% coverage is possible. However, it is not always the most efficient testing strategy to take. Sometimes it is smarter to choose a higher-level strategy that subsumes the testing strategies below it, including node and edge coverage.

3. Question 3

a.


C2.B
C2.D
C2.C
C3.B
C4.C
C5.D
C5.C


C3.A
C2.B
C2.D
C2.C
C1.A
C3.B
C4.C
C5.B
C5.D
C4.A
C5.C

    b. C1 is the superclass and, you can only access methods and member variables of the superclass through that variable. So, to access the “D” method you either need to declare a variable of a concrete class or cast it to a concrete type.

4. Question 4
    a. i == 1
    b. i != 1
    c. any i

References:

JEdit. (2020, October 18). Retrieved November 06, 2020, from https://sourceforge.net/projects/jedit/

Nord, R. (2020, March 16). Using Machine Learning to Detect Design Patterns. Retrieved November 06, 2020, from https://insights.sei.cmu.edu/sei_blog/2020/03/using-machine-learning-to-detect-design-patterns.html

Salxandersalxander 98155 gold badges1818 silver badges2828 bronze badges, AttilaAttila 25.2k33 gold badges3939 silver badges5151 bronze badges, Binariousbinarious 4, User973999user973999, & Raviookraviook 9944 bronze badges. (1961, June 01). Calling a subclass method from superclass. Retrieved November 06, 2020, from https://stackoverflow.com/questions/10021603/calling-a-subclass-method-from-superclass