**DEMOGRAPHIC INFORMATION**

1.- Gender: Male (x) Female ( )

2.- Age: \_\_\_21\_\_\_\_

3.- Studies: PhD ( ) Master ( ) Bachelor ( ) Student (x) None ( )

4.- Years of experience in software development: \_\_\_\_\_\_3\_\_\_\_\_\_\_

5.- Years of experience in Java software development: \_\_\_\_\_1\_\_\_\_\_\_

6.- Rate from 1 (None) to 5 (Expert) your level of knowledge of Java: 1( ) 2( ) 3(x) 4( ) 5( )

**EVALUATION**

**A.** The following Java methods are part of the "compiler" project, which is a compiler for a simple programming language, for use in education.

1. The method is included in the "SymbolTable" class, and has the following code:

**public** **void** \_\_\_\_\_\_\_\_\_\_\_\_\_\_(String lexeme, Symbol.SymbolType kind, ArrayList<String> argumentsDataType, String returnType, **int** numOfVar) {

Symbol symbol = **new** Symbol();

symbol.lexeme = lexeme;

symbol.kind = kind;

symbol.argumentsDataType = argumentsDataType;

symbol.returnType = returnType;

symbol.numOfVar = numOfVar;

symbols.add(symbol);

}

For the following method names, indicate whether they are appropriate for the previous method. Rate from 1 (disagree) to 5 (strongly agree) your level of agreement.

|  |  |  |
| --- | --- | --- |
| **Method name** | **Is the name appropriate?** | **Comments (optional)** |
| symbolTableCompilerMethod | 1(x) 2( ) 3( ) 4( ) 5( ) | The name doesn’t reflect the purpose of the function at all |
| addToSymbolTable | 1( ) 2( ) 3( ) 4( ) 5(x) |  |
| addFunctionSymbolTable | 1( ) 2(x) 3( ) 4( ) 5( ) | It’s incorrect (a variable is also a symbol and it can be added by this method) |
| addSubroutineSymbol | 1( ) 2(x) 3( ) 4( ) 5( ) | Same as the above |

2. The method is included in the "Parser" class, and has the following code:

**private** **void** \_\_\_\_\_\_\_\_\_\_\_\_\_(String vmCodes) {

**try** {

File log = **new** File(lexer.getFolderPath() + File.***separator*** + className + ".vm");

FileWriter fileWriter = **new** FileWriter(log.getAbsoluteFile(), **true**);

BufferedWriter bufferedWriter = **new** BufferedWriter(fileWriter);

bufferedWriter.write(vmCodes);

bufferedWriter.close();

fileWriter.close();

} **catch** (IOException e) {

e.printStackTrace();

}

}

For the following method names, indicate whether they are appropriate for the previous method. Rate from 1 (disagree) to 5 (strongly agree) your level of agreement.

|  |  |  |
| --- | --- | --- |
| **Method name** | **Is the name appropriate?** | **Comments (optional)** |
| vmCodeInput | 1(x) 2( ) 3( ) 4( ) 5( ) |  |
| appendVmCodesToFile | 1( ) 2( ) 3( ) 4(x) 5( ) |  |
| parserMethod | 1(x) 2( ) 3( ) 4( ) 5( ) |  |
| insertVMCodesIntoVMFile | 1( ) 2(x) 3( ) 4( ) 5( ) |  |

**B.** The following Java methods are part of the "jvector" project, which is a pure Java embedded vector search engine, used by DataStax Astra DB and Apache Cassandra.

1. The method is included in the "GraphIndexBuilder" class, and has the following code:

**private** **static** Bits \_\_\_\_\_\_\_\_\_\_\_\_\_(**int** node) {

**return** **new** Bits() {

@Override

**public** **boolean** get(**int** index) {

**return** index != node;

}

@Override

**public** **int** length() {

// length is max node id, which could be larger than size after deletes

**throw** **new** UnsupportedOperationException();

}

};

}

For the following method names, indicate whether they are appropriate for the previous method. Rate from 1 (disagree) to 5 (strongly agree) your level of agreement.

|  |  |  |
| --- | --- | --- |
| **Method name** | **Is the name appropriate?** | **Comments (optional)** |
| createNonSelfBits | 1( ) 2( ) 3( ) 4(x) 5( ) |  |
| createBits | 1( ) 2(x) 3( ) 4( ) 5( ) |  |
| createNotSelfBits | 1( ) 2( ) 3( ) 4( ) 5(x) |  |
| graphIndexBuilderMethod | 1(x) 2( ) 3( ) 4( ) 5( ) |  |

2. The method is included in the "ConcurrentNeighborSet" class, and has the following code:

**void** \_\_\_\_\_\_\_\_\_\_\_\_\_(**int** node, **float** score, **boolean** limitConnections) {

neighborsRef.getAndUpdate(current -> {

NodeArray next = current.copy();

**if** (limitConnections) {

// remove the worst edge to make room for the new one

next.size = *min*(next.size, maxConnections - 1);

}

next.insertSorted(node, score);

**return** next;

});

}

For the following method names, indicate whether they are appropriate for the previous method. Rate from 1 (disagree) to 5 (strongly agree) your level of agreement.

|  |  |  |
| --- | --- | --- |
| **Method name** | **Is the name appropriate?** | **Comments (optional)** |
| updateNeighbors | 1( ) 2( ) 3( ) 4( ) 5(x) |  |
| insertNotDiverse | 1(x) 2( ) 3( ) 4( ) 5( ) |  |
| concurrentNeighborSetMethod | 1(x) 2( ) 3( ) 4( ) 5( ) |  |
| insertNonDiverseNode | 1(x) 2( ) 3( ) 4( ) 5( ) |  |

**C.** The following Java methods are part of the "Log4-detector" project, which is a scanner that detects Log4J vulnerabilities.

1. The method is included in the "Log4JDetector" class, and has the following code:

**private** **static** **boolean** \_\_\_\_\_\_\_\_\_\_\_\_\_(BufferedInputStream in) {

in.mark(4);

**try** {

**int**[] fourBytes = *pop4*(in);

**return** *isZipSentinel*(fourBytes);

} **catch** (IOException ioe) {

**return** **false**;

} **finally** {

**try** {

in.reset();

} **catch** (IOException ioe) {

**throw** **new** RuntimeException("BufferedInputStream.reset() failed: " + ioe);

}

}

}

For the following method names, indicate whether they are appropriate for the previous method. Rate from 1 (disagree) to 5 (strongly agree) your level of agreement.

|  |  |  |
| --- | --- | --- |
| **Method name** | **Is the name appropriate?** | **Comments (optional)** |
| log4JDetectorMethod | 1(x) 2( ) 3( ) 4( ) 5( ) |  |
| startsWithZipMagic | 1( ) 2( ) 3( ) 4( ) 5(x) |  |
| isZipSentinel | 1(x) 2( ) 3( ) 4( ) 5( ) | Will cause nonintended recursion |
| isZipFile | 1( ) 2(x) 3( ) 4( ) 5( ) |  |

2. The method is included in the "Strings" class, and has the following code:

**public** **static** Long \_\_\_\_\_\_\_\_\_\_\_\_\_(String s, **long** defaultVal) {

s = s != **null** ? s.trim() : "";

**if** ("".equals(s)) {

**return** defaultVal;

} **try** {

**return** Long.*parseLong*(s);

} **catch** (RuntimeException e) {

**return** defaultVal;

}

}

For the following method names, indicate whether they are appropriate for the previous method. Rate from 1 (disagree) to 5 (strongly agree) your level of agreement.

|  |  |  |
| --- | --- | --- |
| **Method name** | **Is the name appropriate?** | **Comments (optional)** |
| parseLongWithDefault | 1( ) 2(x) 3( ) 4( ) 5( ) |  |
| stringMethod | 1(x) 2( ) 3( ) 4( ) 5( ) |  |
| parseLongOrDefault | 1( ) 2( ) 3( ) 4( ) 5(x) |  |
| parseLong | 1( ) 2(x) 3( ) 4( ) 5( ) |  |

**D.** The following Java methods are part of the "ramen" project, which implements a simple social network, and was used in a Java programming course.

1. The method is included in the internal class "App" of the class "Main", and has the following code:

**private** **void** \_\_\_\_\_\_\_\_\_\_\_\_\_() {

gMap.clear();

home.removeAllChildren();

tree.reload(home);

**for** (Group g : *app*.listGroups()) {

gTreeNode supernode = g.getSupergroup() == **null** ? home

: gMap.get(g.getSupergroup());

gTreeNode node = **new** gTreeNode(g);

tree.insertNodeInto(node, supernode, supernode.getChildCount());

gMap.put(g, node);

}

**for** (**int** i = 0; i < gTree.getRowCount(); i++)

gTree.expandRow(i);

}

For the following method names, indicate whether they are appropriate for the previous method. Rate from 1 (disagree) to 5 (strongly agree) your level of agreement.

|  |  |  |
| --- | --- | --- |
| **Method name** | **Is the name appropriate?** | **Comments (optional)** |
| refreshTreeView | 1( ) 2( ) 3( ) 4( ) 5(x) |  |
| fetchGroups | 1(x) 2( ) 3( ) 4( ) 5( ) |  |
| appMethod | 1(x) 2( ) 3( ) 4( ) 5( ) |  |
| updateGroupTree | 1( ) 2( ) 3( ) 4(x) 5( ) |  |

2. The method is included in the "Question" class, and has the following code:

**public** List<User> \_\_\_\_\_\_\_\_\_\_\_\_\_() {

Group g = (Group) **this**.getTo(); //**TODO**: Review

List<User> ret = **new** ArrayList<>(g.getMembers());

List<User> ans = **this**.whoAnswered();

**for**(User u: g.getMembers()) {

**if**(ans.contains(u) || u.getClass() == Sensei.**class**) {

ret.remove(u);

}

}

**return** ret;

}

For the following method names, indicate whether they are appropriate for the previous method. Rate from 1 (disagree) to 5 (strongly agree) your level of agreement.

|  |  |  |
| --- | --- | --- |
| **Method name** | **Is the name appropriate?** | **Comments (optional)** |
| getUsersWhoDidNotAnswer | 1( ) 2( ) 3( ) 4( ) 5(x) |  |
| findUsersWhoDidntAnswer | 1( ) 2( ) 3( ) 4(x) 5( ) |  |
| whoDidntAnswer | 1(x) 2( ) 3( ) 4( ) 5( ) |  |
| questionMethod | 1(x) 2( ) 3( ) 4( ) 5( ) |  |