BCGen Questionnaire

We used three anonymous models to generate code comments. The quality of comments generated by each model is mainly measured from two aspects: naturalness and informativeness. Naturalness describes whether the generated comments are fluent and grammatically accurate; informational describes whether the generated comments correctly reflect the bytecode information. We provide the source code as a reference, that is, it is necessary to judge whether the comment correctly reflects the function of the source code. Both naturalness and informativeness are scored on a scale of 0-4:
A. Naturalness
0: The sentence is incomplete, illogical and has grammatical errors.
1: The sentence is basically complete, but not smooth and has grammatical errors.
2: The sentence is relatively complete, basically smooth but with grammatical errors.
3: The sentence is relatively complete, basically smooth and without grammatical errors.
4: Sentences are very complete, smooth and free from grammatical

B.Informativeness
0: Comment error describing the source code function.
1: There are a few descriptions in the comments that correctly reflect the function of the source code, but most of the descriptions are wrong.
2: Most of the descriptions of the comments correctly reflect the source code functions, and a small part of the descriptions are wrong.
3: Comments can basically and accurately describe the function of the source code.
4: Comments can very accurately describe the function of the source code.

1. Basic information collection *

Name	____________________
Age	____________________
Affiliated institution (e.g., XXX university, XXX company, etc.)	____________________
Programming experience （years）	____________________
Programming languages you are at (e.g., Java, C++, Python etc.)	____________________
English level（e.g., CET4, CET6, IELTS, TOEFL etc. ）	____________________

2. comment1（by model 1）: updates the selection of field to exclude when write

comment2（by model 2）: updates the field of field when write the field to be write

comment3（by model 3）: updates the name of the field that write a write a a string

source code:
```
public void updateFieldExclusion(String fieldsToExclude) {
    if (headers == null) {
        throw new IllegalStateException("Cannot de-select fields by name. Headers not defined.");
    }
    internalSettings.excludeFields(fieldsToExclude);
    updateIndexesToWrite(internalSettings);
}
```
*

	0	1	2	3	4
naturalness of comment1	○	○	○	○	○
naturalness of comment2	○	○	○	○	○
naturalness of comment3	○	○	○	○	○
informativeness of comment1	○	○	○	○	○
informativeness of comment2	○	○	○	○	○
informativeness of comment3	○	○	○	○	○

3. comment1（by model 1）: add an validator to the aggregator

comment2（by model 2）: add an validator to the aggregator

comment3（by model 3）: add an validator to the aggregator

source code:

```
public void addValidator(LineValidator validator) {
    if (validator != null) {
        validators.add(validator);
    }
}
```
*

	0	1	2	3	4
naturalness of comment1	○	○	○	○	○
naturalness of comment2	○	○	○	○	○
naturalness of	○	○	○	○	○

comment3					
informativeness of comment1	○	○	○	○	○
informativeness of comment2	○	○	○	○	○
informativeness of comment3	○	○	○	○	○

4. comment1（by model 1）: pushes a node on to the stack

comment2（by model 2）: pushes a node on the stack

comment3（by model 3）: pushes the give node to the node

source code:

```
void pushNode(Node n) {
    nodes.push(n);
    ++sp;
} *
```

	0	1	2	3	4
naturalness of comment1	○	○	○	○	○
naturalness of comment2	○	○	○	○	○
naturalness of comment3	○	○	○	○	○
informativeness of comment1	○	○	○	○	○
informativeness of comment2	○	○	○	○	○
informativeness of	○	○	○	○	○

comment3					

5. comment1（by model 1）: builds the classifier to generate a partition

comment2（by model 2）: builds the classifier to partition

comment3（by model 3）: builds the classifier for a a partition

source code:

```
public void generatePartition(Instances data) throws Exception {
    if (m_Classifier instanceof PartitionGenerator)
       buildClassifier(data);
   else throw new Exception( "Classifier: " + getClassifierSpec() + " cannot generate a partition");
} *
```

	0	1	2	3	4
naturalness of comment1	○	○	○	○	○
naturalness of comment2	○	○	○	○	○
naturalness of comment3	○	○	○	○	○
informativeness of comment1	○	○	○	○	○
informativeness of comment2	○	○	○	○	○
informativeness of comment3	○	○	○	○	○

6. comment1（by model 1）: sorts the specified range of the receiver into ascend order

comment2（by model 2）: sorts the specified range of the receiver into ascend order

comment3（by model 3）: sorts the specified range of the receiver into ascend order

source code:

```
public void mergeSortFromTo(int from, int to) {
   int mySize = size();
   checkRangeFromTo(from, to, mySize);
```

```
    byte[] myElements = elements();
    Sorting.mergeSort(myElements, from, to + 1);
    elements(myElements);
    setSizeRaw(mySize);
} *
```

	0	1	2	3	4
naturalness of comment1	○	○	○	○	○
naturalness of comment2	○	○	○	○	○
naturalness of comment3	○	○	○	○	○
informativeness of comment1	○	○	○	○	○
informativeness of comment2	○	○	○	○	○
informativeness of comment3	○	○	○	○	○

7. comment1（by model 1）: sets the secure random generator configuration

comment2（by model 2）: sets the secure random generator

comment3（by model 3）: sets the secure random generator configuration

source code:

```
public void setSecureRandom(SecureRandomFactoryBean secureRandom) {
    this.secureRandom = secureRandom;
} *
```

	0	1	2	3	4
naturalness of comment1	○	○	○	○	○
naturalness of comment2	○	○	○	○	○

naturalness of comment3	○	○	○	○	○
informativeness of comment1	○	○	○	○	○
informativeness of comment2	○	○	○	○	○
informativeness of comment3	○	○	○	○	○

8. comment1（by model 1）: remove a class from the class cache

comment2（by model 2）: delete a class entry from the cache

comment3（by model 3）: remove entry param from the class param classname the classname to to to be cache

source code:

```
protected void removeClassCacheEntry(String name) {
    classCache.remove(name);
} *
```

	0	1	2	3	4
naturalness of comment1	○	○	○	○	○
naturalness of comment2	○	○	○	○	○
naturalness of comment3	○	○	○	○	○
informativeness of comment1	○	○	○	○	○
informativeness of comment2	○	○	○	○	○
informativeness of comment3	○	○	○	○	○

9. comment1（by model 1）: write byte from the specify byte array to the stream

comment2（by model 2）: write byte byte to the underlie data byte array

comment3（by model 3）: write the byte bytes to the specify byte array param data the byte to start param srcoffset the offset param length the length of byte to throw ioexception if an error occurs

source code:

```
public void write(final byte[] b, int off, int len) throws IOException {
    while (len > 0) {
        final int c = Math.min(len, decoderIn.remaining());
        decoderIn.put(b, off, c);
        processInput(false);
        len -= c; off += c;
    }
    if (writeImmediately) {
        flushOutput();
    }
}
```
*

	0	1	2	3	4
naturalness of comment1	○	○	○	○	○
naturalness of comment2	○	○	○	○	○
naturalness of comment3	○	○	○	○	○
informativeness of comment1	○	○	○	○	○
informativeness of comment2	○	○	○	○	○
informativeness of comment3	○	○	○	○	○

10. comment1（by model 1）: parse input stream to create xml document

comment2（by model 2）: add an validator to the aggregator

comment3（by model 3）: parse xml document to create xml document source code:

```
public XmlDocument parseInputStream(InputStream is) throws XmlBuilderException {
    XmlPullParser pp = null;
    try {
        pp = factory.newPullParser();
        pp.setInput(is, null); //set options ...
    } catch (XmlPullParserException e) {
        throw new XmlBuilderException("could not start parsing input stream", e);
    } return parse(pp);
}
```
*

	0	1	2	3	4
naturalness of comment1	○	○	○	○	○
naturalness of comment2	○	○	○	○	○
naturalness of comment3	○	○	○	○	○
informativeness of comment1	○	○	○	○	○
informativeness of comment2	○	○	○	○	○
informativeness of comment3	○	○	○	○	○

11. comment1（by model 1）: increment the evaluation count

comment2（by model 2）: increment the number of evaluation evaluation

comment3（by model 3）: increment the counter in evaluation evaluation

source code:

```
protected void incrementEvaluationCount() throws TooManyEvaluationsException {
    evaluations.incrementCount();
}
```
*

	0	1	2	3	4

naturalness of comment1	○	○	○	○	○
naturalness of comment2	○	○	○	○	○
naturalness of comment3	○	○	○	○	○
informativeness of comment1	○	○	○	○	○
informativeness of comment2	○	○	○	○	○
informativeness of comment3	○	○	○	○	○

12. comment1（by model 1）: create a groupmatcher that match job group start with the give string

comment2（by model 2）: create a groupmatcher that match match group start with the give string

comment3（by model 3）: create a groupmatcher that match job group start with the give string

source code:

```
public static GroupMatcher<JobKey> jobGroupStartsWith(String compareTo) {
    return GroupMatcher.groupStartsWith(compareTo);
}
```
*

	0	1	2	3	4
naturalness of comment1	○	○	○	○	○
naturalness of comment2	○	○	○	○	○
naturalness of comment3	○	○	○	○	○
informativeness of comment1	○	○	○	○	○
informativene	○	○	○	○	○

ss of comment2					
informativeness of comment3	○	○	○	○	○

13. comment1（by model 1）: replaces all match within the builder with the replace string

comment2（by model 2）: replaces all part of the builder with the replace string

comment3（by model 3）: replaces the first match within the builder in the builder

source code:
```
public StrBuilder replaceAll(final StrMatcher matcher, final String replaceStr) {
    return replace(matcher, replaceStr, 0, size, -1);
} *
```

	0	1	2	3	4
naturalness of comment1	○	○	○	○	○
naturalness of comment2	○	○	○	○	○
naturalness of comment3	○	○	○	○	○
informativeness of comment1	○	○	○	○	○
informativeness of comment2	○	○	○	○	○
informativeness of comment3	○	○	○	○	○

14. comment1（by model 1）: returns the tip text for this property

comment2（by model 2）: returns the string of this this tool tip text this tool

comment3（by model 3）: enables the text to text for this use when create a tool text

source code:
```
public String charSetTipText() {
    return "The character set to use when reading text files (eg UTF-8) - leave" + " blank to use the default character set.";
} *
```

	0	1	2	3	4
naturalness of comment1	○	○	○	○	○
naturalness of comment2	○	○	○	○	○
naturalness of comment3	○	○	○	○	○
informativeness of comment1	○	○	○	○	○
informativeness of comment2	○	○	○	○	○
informativeness of comment3	○	○	○	○	○

15. comment1（by model 1）: use the default visual appearance

comment2（by model 2）: use the default visual appearance

comment3（by model 3）: use the default visual appearance for this plot

source code:

```
public void useDefaultVisual() {
    m_visual.loadIcons(BeanVisual.ICON_PATH + "DefaultDataVisualizer.gif", BeanVisual.ICON_PATH + "DefaultDataVisualizer_animated.gif");
} *
```

	0	1	2	3	4
naturalness of comment1	○	○	○	○	○
naturalness of comment2	○	○	○	○	○
naturalness of	○	○	○	○	○

comment3					
informativeness of comment1	○	○	○	○	○
informativeness of comment2	○	○	○	○	○
informativeness of comment3	○	○	○	○	○

16. comment1（by model 1）: sets the allowed url scheme

comment2（by model 2）: sets the allowed url scheme

comment3（by model 3）: sets the scheme parameters to use

source code:

```
public void setPermittedURISchemes(final URIScheme[] permittedURISchemes) {
    this.permittedURISchemes.clear();
    ArrayUtils.addAll(this.permittedURISchemes, permittedURISchemes);
} *
```

	0	1	2	3	4
naturalness of comment1	○	○	○	○	○
naturalness of comment2	○	○	○	○	○
naturalness of comment3	○	○	○	○	○
informativeness of comment1	○	○	○	○	○
informativeness of comment2	○	○	○	○	○
informativeness of	○	○	○	○	○

comment3					

17. comment1（by model 1）: set the seed for random number generation

comment2（by model 2）: set the seed for the random number generator

comment3（by model 3）: sets the random seed generator seed value

source code:

```
public void setSeed(int value) {
    m_Seed = value;
} *
```

	0	1	2	3	4
naturalness of comment1	○	○	○	○	○
naturalness of comment2	○	○	○	○	○
naturalness of comment3	○	○	○	○	○
informativeness of comment1	○	○	○	○	○
informativeness of comment2	○	○	○	○	○
informativeness of comment3	○	○	○	○	○

18. comment1（by model 1）: set the batch size for process refresh request

comment2（by model 2）: set the size size

comment3（by model 3）: sets batch size

source code:

```
public ExecutionHints setResultBatchSize(int size) {
    this.batchSize = size;
    return this;
} *
```

	0	1	2	3	4
naturalness of comment1	○	○	○	○	○
naturalness of comment2	○	○	○	○	○
naturalness of comment3	○	○	○	○	○
informativeness of comment1	○	○	○	○	○
informativeness of comment2	○	○	○	○	○
informativeness of comment3	○	○	○	○	○

19. comment1（by model 1）: returns an array contain the wire format representation of the message with the specify maximum length

comment2（by model 2）: returns a array contain the wire representation of the wire format

comment3（by model 3）: returns an array contain the wire format representation of the message

source code:

```
public byte[] toWire() {
    DNSOutput out = new DNSOutput();
    toWire(out); size = out.current();
    return out.toByteArray();
} *
```

	0	1	2	3	4
naturalness of comment1	○	○	○	○	○
naturalness of comment2	○	○	○	○	○
naturalness of comment3	○	○	○	○	○

informativeness of comment1	○	○	○	○	○
informativeness of comment2	○	○	○	○	○
informativeness of comment3	○	○	○	○	○

20. comment1（by model 1）: returns the average of two number

comment1（by model 2）: average average of the value value param degree param value param value

comment1（by model 3）: average average of the give value

source code:

```
public static double average(double x1, double x2) {
    return (x1 + x2) / 2.0;
} *
```

	0	1	2	3	4
naturalness of comment1	○	○	○	○	○
naturalness of comment2	○	○	○	○	○
naturalness of comment3	○	○	○	○	○
informativeness of comment1	○	○	○	○	○
informativeness of comment2	○	○	○	○	○
informativeness of comment3	○	○	○	○	○

21. comment1（by model 1）: return the time in millisecond since the last notification

comment2（by model 2）: return the linear price grid of the { link # }

comment3（by model 3）: return the time in millisecond the the long time in the

source code:

```
public long scheduledTime() {
    return callable.getScheduled();
} *
```

	0	1	2	3	4
naturalness of comment1	○	○	○	○	○
naturalness of comment2	○	○	○	○	○
naturalness of comment3	○	○	○	○	○
informativeness of comment1	○	○	○	○	○
informativeness of comment2	○	○	○	○	○
informativeness of comment3	○	○	○	○	○

22. comment1（by model 1）: answers the number of element in the subgroup

comment2（by model 2）: answers the number of thread

comment3（by model 3）: answers the total number of subgroup

source code:

```
public int getSubgroupCount() {
    return subgroupCount;
} *
```

	0	1	2	3	4
naturalness of comment1	○	○	○	○	○

naturalness of comment2	○	○	○	○	○
naturalness of comment3	○	○	○	○	○
informativeness of comment1	○	○	○	○	○
informativeness of comment2	○	○	○	○	○
informativeness of comment3	○	○	○	○	○

23. comment1（by model 1）: gets the error message for this query failure

comment2（by model 2）: gets the error message

comment3（by model 3）: get the error message

source code:

```
public String getErrorMessage() {
    return errorMsg;
} *
```

	0	1	2	3	4
naturalness of comment1	○	○	○	○	○
naturalness of comment2	○	○	○	○	○
naturalness of comment3	○	○	○	○	○
informativeness of comment1	○	○	○	○	○
informativeness of comment2	○	○	○	○	○

informativeness of comment3	○	○	○	○	○

24. comment1（by model 1）: find the default node value

comment2（by model 2）: find the node value of the property value

comment3（by model 3）: find the default default value

source code:

```
public static boolean getNodeDefaultVoid() {
   return booleanValue("NODE_DEFAULT_VOID");
} *
```

	0	1	2	3	4
naturalness of comment1	○	○	○	○	○
naturalness of comment2	○	○	○	○	○
naturalness of comment3	○	○	○	○	○
informativeness of comment1	○	○	○	○	○
informativeness of comment2	○	○	○	○	○
informativeness of comment3	○	○	○	○	○

25. comment1（by model 1）: returns the class distribution for an instance

comment2（by model 2）: computes the distribution distribution with a set of instance with the give instance

comment3（by model 3）: build the give instance

source code:

```
public double [] distributionForInstance(Instance instance) throws Exception {
    double[] probOfClassGivenDoc = new double[m_numClasses];
    double[] logDocGivenClass = new double[m_numClasses];
```

```
  for (int h = 0; h < m_numClasses; h++) {
    logDocGivenClass[h] = probOfDocGivenClass(instance, h);
  }
  double max = logDocGivenClass[Utils.maxIndex(logDocGivenClass)];
  for (int i = 0; i < m_numClasses; i++) {
    probOfClassGivenDoc[i] = Math.exp(logDocGivenClass[i] - max) * m_probOfClass[i];
  }
  Utils.normalize(probOfClassGivenDoc);
  return probOfClassGivenDoc;
}
```
*

	0	1	2	3	4
naturalness of comment1	○	○	○	○	○
naturalness of comment2	○	○	○	○	○
naturalness of comment3	○	○	○	○	○
informativeness of comment1	○	○	○	○	○
informativeness of comment2	○	○	○	○	○
informativeness of comment3	○	○	○	○	○

26. comment1（by model 1）: return the system temp directory

comment2（by model 2）: return the system temp return the temp directory

comment3（by model 3）: return the system system system property return system temp directory

source code:

```
public static String getTempDirectoryPath() {
   return System.getProperty("java.io.tmpdir");
}
```
*

	0	1	2	3	4

naturalness of comment1	○	○	○	○	○
naturalness of comment2	○	○	○	○	○
naturalness of comment3	○	○	○	○	○
informativeness of comment1	○	○	○	○	○
informativeness of comment2	○	○	○	○	○
informativeness of comment3	○	○	○	○	○

27. comment1（by model 1）: returns hash code for this immutable object

comment2（by model 2）: an an hashcode for an an array

comment3（by model 3）: gets the hash code for the message

source code:

```
public final int hashCode() {
    return id;
} *
```

	0	1	2	3	4
naturalness of comment1	○	○	○	○	○
naturalness of comment2	○	○	○	○	○
naturalness of comment3	○	○	○	○	○
informativeness of comment1	○	○	○	○	○
informativene	○	○	○	○	○

ss of comment2					
informativeness of comment3	○	○	○	○	○

28. comment1（by model 1）: computes the dot product of the double value

comment2（by model 2）: returns the product of the counter

comment3（by model 3）: return the product of the following 2

source code:

```
public static double dotProduct(double[] a, double[] b) {
    if (a.length != b.length) {
        throw new RuntimeException("Can\'t calculate dot product of multiple different lengths: a.length=" + a.length + "
b.length=" + b.length);
    }
    double result = 0;
    for (int i = 0; i < a.length; i++) {
        result += a[i] * b[i];
    }
    return result;
} *
```

	0	1	2	3	4
naturalness of comment1	○	○	○	○	○
naturalness of comment2	○	○	○	○	○
naturalness of comment3	○	○	○	○	○
informativeness of comment1	○	○	○	○	○
informativeness of comment2	○	○	○	○	○
informativene	○	○	○	○	○

ss of comment3					

29. comment1（by model 1）: returns whether rolloverproducer be enable

comment2（by model 2）: returns true if this table be enable with the main field

comment3（by model 3）: < p > overridden to return true if the component be disabled

source code:

```
public boolean isRolloverEnabled() {
    return rolloverProducer != null;
} *
```

	0	1	2	3	4
naturalness of comment1	○	○	○	○	○
naturalness of comment2	○	○	○	○	○
naturalness of comment3	○	○	○	○	○
informativeness of comment1	○	○	○	○	○
informativeness of comment2	○	○	○	○	○
informativeness of comment3	○	○	○	○	○

30. comment1（by model 1）: clear the memory

comment2（by model 2）: reads a 32 back from the memory area

comment3（by model 3）: removes the first of 32 bit from the give number

source code:

```
public IRubyObject clear(ThreadContext context) {
    getMemoryIO().setMemory(0, size, (byte) 0);
    return this;
} *
```

	0	1	2	3	4
naturalness of comment1	○	○	○	○	○
naturalness of comment2	○	○	○	○	○
naturalness of comment3	○	○	○	○	○
informativeness of comment1	○	○	○	○	○
informativeness of comment2	○	○	○	○	○
informativeness of comment3	○	○	○	○	○

31. comment1（by model 1）: sets the url use by the database connection

comment2（by model 2）: set the url to be use

comment3（by model 3）: sets a url to load from the loader

source code:

```
public void setUrl(String url) {
    checkEnv();
    m_URL = url;
    String uCopy = m_URL;
    try {
        uCopy = m_env.substitute(uCopy);
    } catch (Exception ex) { }
    m_DataBaseConnection.setDatabaseURL(uCopy);
} *
```

	0	1	2	3	4
naturalness of comment1	○	○	○	○	○
naturalness of comment2	○	○	○	○	○

naturalness of comment3	○	○	○	○	○
informativeness of comment1	○	○	○	○	○
informativeness of comment2	○	○	○	○	○
informativeness of comment3	○	○	○	○	○

32. comment1（by model 1）: return the imaginary part of the eigenvalue

comment2（by model 2）: return the imaginary of of the eigenvalue return the owning vector

comment3（by model 3）: return the imaginary part of the eigenvalue return imag diag d

source code:

```
public Vector getImagEigenvalues() {
    return e;
} *
```

	0	1	2	3	4
naturalness of comment1	○	○	○	○	○
naturalness of comment2	○	○	○	○	○
naturalness of comment3	○	○	○	○	○
informativeness of comment1	○	○	○	○	○
informativeness of comment2	○	○	○	○	○
informativeness of comment3	○	○	○	○	○

33. comment1（by model 1）: detaches this adapter from the wrapped connection

comment2（by model 2）: called when the have be synchronize <SEG> subclass

comment3（by model 3）: detaches this connection

source code:

```
protected synchronized void detach() {
    wrappedConnection = null;
    duration = Long.MAX_VALUE;
} *
```

	0	1	2	3	4
naturalness of comment1	○	○	○	○	○
naturalness of comment2	○	○	○	○	○
naturalness of comment3	○	○	○	○	○
informativeness of comment1	○	○	○	○	○
informativeness of comment2	○	○	○	○	○
informativeness of comment3	○	○	○	○	○

34. comment1（by model 1）: extracts the day of month of the give date

comment2（by model 2）: extracts the day of the give date

comment3（by model 3）: extracts the day of the give date

source code:

```
public static int dayOfMonthOf(Date date) {
    return toCalendar(date).get(Calendar.DAY_OF_MONTH);
} *
```

	0	1	2	3	4

naturalness of comment1	○	○	○	○	○
naturalness of comment2	○	○	○	○	○
naturalness of comment3	○	○	○	○	○
informativeness of comment1	○	○	○	○	○
informativeness of comment2	○	○	○	○	○
informativeness of comment3	○	○	○	○	○

35. comment1（by model 1）: gets whether the collection be cap

comment2（by model 2）: gets whether the collection be be

comment3（by model 3）: return whether the cap be cap

source code:

```
public boolean isCapped() {
    return capped;
} *
```

	0	1	2	3	4
naturalness of comment1	○	○	○	○	○
naturalness of comment2	○	○	○	○	○
naturalness of comment3	○	○	○	○	○
informativeness of comment1	○	○	○	○	○
informativene	○	○	○	○	○

ss of comment2					
informativeness of comment3	○	○	○	○	○

36. comment1（by model 1）: get the tool tip text for this perspective

comment2（by model 2）: get the tool tip text for this perspective

comment3（by model 3）: returns the tip text for this property

source code:

```
public String getPerspectiveTipText() {
    return "Matrix of attribute summary histograms";
} *
```

	0	1	2	3	4
naturalness of comment1	○	○	○	○	○
naturalness of comment2	○	○	○	○	○
naturalness of comment3	○	○	○	○	○
informativeness of comment1	○	○	○	○	○
informativeness of comment2	○	○	○	○	○
informativeness of comment3	○	○	○	○	○

37. comment1（by model 1）: get the response time to send the response

comment2（by model 2）: the total amount response time in response

comment3（by model 3）: get the response time

source code:

```
public int getResponseSendTime() {
    return responseSendTime;
} *
```

	0	1	2	3	4
naturalness of comment1	○	○	○	○	○
naturalness of comment2	○	○	○	○	○
naturalness of comment3	○	○	○	○	○
informativeness of comment1	○	○	○	○	○
informativeness of comment2	○	○	○	○	○
informativeness of comment3	○	○	○	○	○

38. comment1（by model 1）: sets the name of the bucket involve in the original request

comment2（by model 2）: sets the name of the amazon s3 bucket contain the object s

comment3（by model 3）: sets the name of the amazon s3 bucket contain the object s to delete

source code:

```
public void setBucketName(String bucketName) {
    this.bucketName = bucketName;
} *
```

	0	1	2	3	4
naturalness of comment1	○	○	○	○	○
naturalness of comment2	○	○	○	○	○
naturalness of comment3	○	○	○	○	○
informativene	○	○	○	○	○

ss of comment1					
informativeness of comment2	○	○	○	○	○
informativeness of comment3	○	○	○	○	○

39. comment1（by model 1）: encodes an object use the give object

comment2（by model 2）: encodes an object use the refined soundex algorithm

comment3（by model 3）: encodes an object into it url safe form

source code:

```
public final Object encode(final Object pObject) throws EncoderException {
    if (!(pObject instanceof String)) {
        throw new EncoderException( "Parameter supplied to Match Rating Approach encoder is not of type 
java.lang.String");
    }
    return encode((String) pObject);
}
```
*

	0	1	2	3	4
naturalness of comment1	○	○	○	○	○
naturalness of comment2	○	○	○	○	○
naturalness of comment3	○	○	○	○	○
informativeness of comment1	○	○	○	○	○
informativeness of comment2	○	○	○	○	○
informativeness of	○	○	○	○	○

comment3					

40. comment1（by model 1）: returns true if this operation be timeout

comment2（by model 2）: indicate that the tcp timeout timeout value have be out

comment3（by model 3）: indicates whether the timeout be timeout

source code:

```
public boolean isTimeout() {
    return timeout;
} *
```

	0	1	2	3	4
naturalness of comment1	○	○	○	○	○
naturalness of comment2	○	○	○	○	○
naturalness of comment3	○	○	○	○	○
informativeness of comment1	○	○	○	○	○
informativeness of comment2	○	○	○	○	○
informativeness of comment3	○	○	○	○	○

41. comment1（by model 1）: guarantees that all the len be actually read off the transport

comment2（by model 2）: guarantees that all of byte len transport be actually read off the transport

comment3（by model 3）: read up to len many byte into buf at offset param buf house bytes read param off offset into buff to begin write to param len maximum number of byte to read return number of byte actually read

source code:

```
public int readAll(byte[] buf, int off, int len) throws TTransportException {
    int got = 0;
    int ret = 0;
    while (got < len) {
            ret = read(buf, off+got, len-got);
            if (ret < 0) {
                throw new TTransportException("Error in reading from file");
            }
            if(ret == 0) {
                 throw new TTransportException(TTransportException.END_OF_FILE, "End of File reached");
            }
            got += ret;
        }
        return got;
    } *
```

	0	1	2	3	4
naturalness of comment1	○	○	○	○	○
naturalness of comment2	○	○	○	○	○
naturalness of comment3	○	○	○	○	○
informativeness of comment1	○	○	○	○	○
informativeness of comment2	○	○	○	○	○
informativeness of comment3	○	○	○	○	○