A Java Programming Learning Assistant System Using Test-Driven Development Method

Nobuo Funabiki, Yukiko Matsushima, Toru Nakanishi, Kan Watanabe, and Noriki Amano

***Abstract*—Recently, the object-oriented programming language *Java* has been used in many practical systems including enterprise servers, smart phones, and embedded systems, due to its high safety and portability. Thus, a lot of educational institutes have offered Java programming courses to foster Java engineers. In this paper, we propose a Web-based *Java Programming Learning Assistant System (JPLAS)* using the *test-driven development (TDD) method*, to enhance educational effects of Java programming by assisting self-studies of students who have studied the basic Java grammar while reducing teacher loads. In JPLAS, a teacher first registers a Java programming assignment with a statement, a model source code, and a test code. Then, a student writes a source code by reading the statement and the test code, such that the source code can be tested automatically at the server by using *JUnit*, a tool for the TDD method. We confirm the effectiveness of JPLAS through experimental applications to students in our department.**

***Index Terms*—Web system, programming language, Java, test-driven development method, JUnit, code reading**

I. Introduction

With penetrations of the *information and communication technology (ICT)* into our societies, adverse affects of failures of computer and network systems due to software bugs have become intolerable. They sometimes have stopped functions of critical infrastructures such as railways [1], airlines [2], and large banking systems [3]. Along this trend, the software test has been regarded as a last crucial process to avoid productions of software bugs. Then, a *test-driven development (TDD) method* has been focused as an effective software development method that can avoid bugs by writing and testing source codes at the same time [4]. In the TDD method, a test code should be written before writing a source code. A *test code* is a program code to verify the correctness of the outputs from the methods that are implemented in the *source code*.

Java is a useful and practical object-oriented programming language that has been used in a lot of important practical systems including enterprise servers, smart phones, and embedded systems due to its high safety and portability. Thus, Java programming educations have become important to foster professional Java engineers, and many educational institutes over the world have actually offered Java programming courses. In a Java programming course, usually one or a few teachers educate a lot of students at the same time. Because a student needs writing various Java codes by him/herself to master Java programming, a teacher usually gives a lot of Java programming assignments to the students in the class. Then, for a teacher, the verification of Java codes from students and the feedback with proper comments may take an intolerably long time. As a result, some students may miss chances of improving Java programming skills and lose interests in studying Java programming.

In this paper, we propose a Web-based *Java Programming Learning Assistant System (JPLAS)* using the TDD method, to enhance educational effects in Java programming courses in universities or vocational schools by allowing self-studies of students, while reducing teacher loads. By accessing to JPLAS from a Web browser, a student can repeat the *learning cycle* of Java programming until he/she can complete the correct code for each assignment given by a teacher. This cycle consist of 1) reading the test code written by the teacher, 2) writing/modifying the source code, and 3) testing the source code and suggesting the errors if there exist. By repeating this cycle, a student can master Java programming. As a target user of JPLAS, we consider a student who has studied the basic Java grammar in a class and has written simple Java codes in textbooks through exercises, but who may not be able to write a proper code that satisfies the requirements of an assignment described by sentences and/or to properly read source codes written by other persons.

In JPLAS, a teacher first registers a Java programming assignment with a statement, a model source code, and a test code. Then, a student writes a source code by reading the statement and the test code such that the source code can be tested automatically at the server by using a software tool for the TDD method called *JUnit* [5]. Because the Java source code is tested automatically every time it is submitted to the server, a student can keep modifying the code until completing the correct one.

A main feature of JPLAS in Java programming educations is the utilization of the TDD method. In JPLAS, a teacher must disclose the test code to students for each assignment. This means that the teacher can clarify the requirements in the source code that he/she intends for this assignment by describing them in the test code. In fact, a program code including a test code can sometimes be clearer than a description using sentences describing the specifications. In addition, students can study reading existing codes through the test code reading. Reading codes is actually very important to improve programming skills, and can often happen in the real world. Furthermore, JPLAS provides the *error code highlighting function* to help students to find the faults in their source codes by graphically highlighting the corresponding lines [6]. Therefore, using JPLAS, we expect that students become more aggressive in studying Java programming.

The rest of this paper is organized as follows: Section II introduces the TDD method. Section III describes the plat-

Manuscript received Jan. 20, 2013; revised Jan. 20, 2013.

N. Funabiki, Y. Matsushima, T. Nakanishi, and K. Watanabe are with the Department of Electrical and Communication Engineering, Okayama University, Okayama 700-8530, Japan, e-mail: {funabiki,nakanisi,can}@cne.okayama-u.ac.jp.

N. Amano is with the Center for Faculty Development, Okayama University, Okayama 700-8530, Japan, e-mail: amano@cc.okayama-u.ac.jp.

form for JPLAS. Sections IV presents the service functions in JPLAS. Section V shows the evaluation result of JPLAS. Section VI discusses some related works. Section VII concludes this paper with future works.

II. Test-driven Development Method

In this section, we introduce the *test-driven development (TDD) method* with its features.

A. Outline of TDD Method

In the TDD method, the test code should be written before the source code is written, so that it can verify whether the source code satisfies the required specifications during its development process. The basic code development cycle in the TDD method is as follows:

(1) to write a test code that can test every specification,
(2) to write a source code, and
(3) to repeat modifications of the source code until it passes every test by the test code.

B. JUnit

In JPLAS, we adopt *JUnit (JUnit4)* as an open-source Java framework to support the TDD method. *JUnit* can assist a unit test of a Java code unit or a *class*. Because *JUnit* has been designed with the Java-user friendly style, its use including a test code programming is easy for Java programmers. In *JUnit*, a test of a code is performed by using a method whose name starts from "assert". For example, this paper adopts the "assertEquals" method to compare the execution result of the source code with its expected value.

C. Test Code

A test code should be written using libraries provided in *JUnit*. By using the *Math* class source code, we explain how to write a test code. The *Math* class returns the summation of the two integer arguments.

```
1: public class Math{
2:     public int plus(int a, int b){
3:         return( a + b );
4:     }
5: }
```

Then, the following test code can test the *plus* method in the *Math* class.

```
1: import static org.junit.Assert.*;
2: import org.junit.Test;
3: public class MathTest {
4:     @Test
5:     public void testPlus(){
6:        Math ma = new Math();
7:        int result = ma.plus(1, 4);
8:        asserEquals(5, result);
9:    }
10:}
```

This test code imports *JUnit* packages at lines 1 and 2, and declares *MathTest* at line 3. *@Test* at line 4 indicates that the succeeding method represents the test method. Then, it describes the test method *testPlus* to test *plus* in *Math* by the following steps:

(1) to generate an instance for the *Math* class,
(2) to call a method in the instance in (1) using the given arguments, and
(3) to compare the result with its expected value for the arguments in (2) using the *assertEquals* method.

D. Features in TDD Method

In the TDD method, the following features can be observed:

1) The test code can represent the specifications of a program, where it must describe any function to be tested in the program.
2) The test code can be useful in considering the program structure.
3) The test process of a source code becomes efficient, because each function can be tested individually.
4) The refactoring process of a source code becomes easy, because the modified code can be tested instantly.

Thus, to study the test code description is useful for students studying Java programming, because the test code can be equivalent to the program specification. Beside, students should experience the software test that has become important in producing practical software. In future studies, we will put a learning function for test code descriptions into practical use in JPLAS.

III. Platform for JPLAS

In this section, we describe the software platform for JPLAS.

A. Server Platform

JPLAS is implemented using *JSP/Servlet* with *Java 1.6.2* as a Web application on a server, where it adopts the operating system *Ubuntu Server 10.04*, the Web application server *Tomcat 6.0.26*, and the database system *MySQL 5.0.27*, as shown in Figure 1.

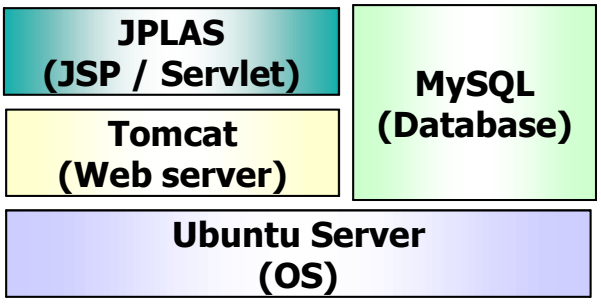

Fig. 1. Server platform.

B. Secure Testing Environment

A source code from a student may contain defective commands such as an infinite loop and an illegal file access. To protect the server from them, two methods, namely a *code analysis* and an *execution time monitoring*, are adopted in JPLAS. The *code analysis* analyzes the source code form

a student to find whether it contains commands that may give adverse effects to the server. Specifically, if it contains a class for the file access such as "Buffered Write" and "PrintWriter", and a class for using external commands such as "Process" and "Runtime", the test execution for this code is aborted and the test failure is returned to the student. The *execution time monitoring* runs the source code on a thread so that the execution time is observed from the main program as shown in Figure 2. If the execution time exceeds a given limit or an exception is detected from the source code, the test is aborted and the failure is returned.

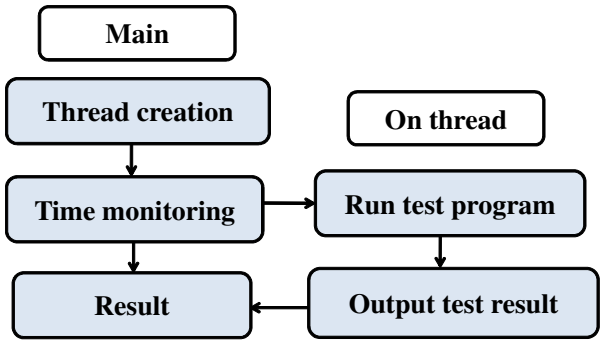

Fig. 2. Execution time monitoring by thread for Java code test.

IV. Service Functions in JPLAS

In this section, we present the service functions implemented in JPLAS, which consist of teacher service functions and student service functions.

A. Teacher Service Functions

Teacher service functions include the registration of new classes, the registration and management of assignments, and the testing of source codes submitted from students. To register a new assignment, a teacher needs to submit an assignment title, a problem statement, a model source code, and a test code to JPLAS. They will be disclosed to the students except for the model source code. Note that the test code must pass the model code correctly.

Using the following correspondence between a source code and a test code, JPLAS automatically generates a template for the test code from the model source code:

- The class name is given by the *test class name* + *Test*.
- The method name is given by the *test* + *test method name*.

Thus, a teacher only needs to specify the specific values for the arguments in each test method to complete the test code, as shown in Figure 3.

To evaluate the difficulty of assignments and comprehensions of students, JPLAS allows a teacher to view the number of submissions for code testing by each student. If a teacher finds that a lot of resubmissions have been done by many students for an assignment, this assignment can be considered too difficult for them and should be changed to an easier one. If a teacher finds a student who submitted codes many times whereas other students did so fewer times, this student should be cared extraordinarily.

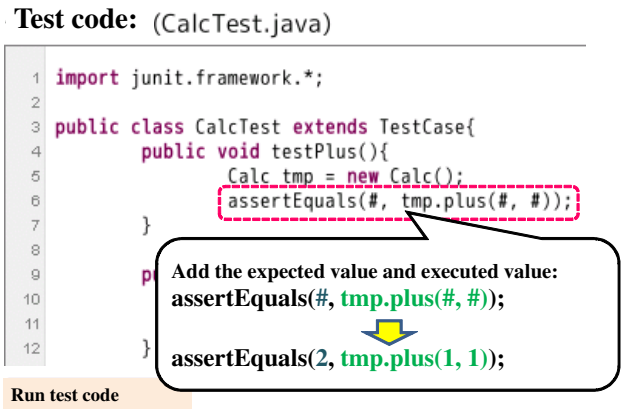

Fig. 3. Test code template.

B. Student Service Functions

Student service functions include the view of the assignments, the submission of source codes for assignments, and the feedback from source code tests at the server. A student should write a source code for an assignment by reading the problem statement and the test code, where he/she must use the class/method names, the types, and the argument setting that are specified in the test code. JPLAS implements a Web-based source code editor called *CodePress* [7] so that a student can write codes on a Web browser. The submitted source codes are stored in the database at the server so that they can view old ones.

As an example, Figures 4 and 5 show a test code from a teacher and a source code from a student for the *ElGamal* encryption programming assignment. Figure 6 shows the test result, where one error is detected because the last argument of the "encrypt" method in the source code is different from the specification given in the test code.

As a feedback function on the source code test from the server, JPLAS implements the *error code highlighting function* to help students to debug their source codes, in addition to displaying the output log of *JUnit*. The *JUnit* log may be hard for these students who even cannot find the lines that they need to modify in their source codes. Thus, we implement a function of highlighting the lines that contain the error codes found by *JUnit*. This function actually highlights both the lines in the test code returning errors at the test and the corresponding lines in the source code, as discussed in the following subsections.

1) Highlighting in Test Code: This function highlights the lines containing the test methods such as *assertEquals* returning errors in the test code. In our implementation, these lines are extracted from the *JUnit* output log that contains the erroneous line information. Figure 7 shows an example JUnit output log, where the line framed in by a rectangular indicates that the 10th line in the test code returns an error. We note that the 24th line also returns an error, which is omitted in Figure 7 to save space. Figure 8 shows the corresponding interface to students for this test code highlighting.

2) Highlighting in Source Code: Then, this function highlights the lines in the source code that declare the methods containing the erroneous lines found by *JUnit*. These erro-

```
import junit.framework.*;

public class EncryptionTest extends TestCase{
        public void testEncrypt(){
                Encryption tmp = new Encryption();

                String[] ciphertext = new String[2];
                ciphertext[0] = "8396436347439611471150025544156611861856194156
                ciphertext[1] = "4294111618369885643725059759348432594977520614
                String[] answer = new String[2];
                answer = tmp.encrypt("10753419069985177098668631553298165241184
                                     "2866444882128996767048467429129036294605435677
                                     "8172276160992595031217378031951847886365097356
                                     "7523026006038088529786911260744316104463319722
                                     "20091225");

                assertEquals( ciphertext[0], answer[0]);
                assertEquals( ciphertext[1], answer[1]);
        }

}
```

Five string arguments for "encrypt" method

Fig. 4. Test code from teacher.

```
import java.math.BigInteger;
public class Encryption{
        String[] encrypt(String p, String g, String r, String Y, String M,
String[] ciphertext){
                BigInteger P = new BigInteger(p);
                BigInteger G = new BigInteger(g);
                BigInteger R = new BigInteger(r);
                BigInteger y = new BigInteger(Y);
                BigInteger m = new BigInteger(M);
                BigInteger C1 = new BigInteger("0");
                BigInteger C2 = new BigInteger("0");
                // エルガマルの計算
                // C1 = MY^r mod p, C2 = g^r mod p
                C1 = m.multiply(y.modPow(R, P));
                C2 = G.modPow(R, P);
                ciphertext[0] = C1.toString();
                ciphertext[1] = C2.toString();
                return ciphertext;
        }
}
```

ElGamal encryption

Fig. 5. Source code from student.

neous methods in the source code are extracted from the codes for the methods returning errors in the test code that are found for the test code highlighting. Figure 9 shows an example of the source code highlighting corresponding to Figure 8.

V. Evaluation

To evaluate the effectiveness of JPLAS in the Java programming education, we applied it to students in our department.

A. Assignment for Evaluation

We prepared an assignment of writing a Java code to calculate the area of a circle and a rectangle, and gave it to 42 sophomore students taking the Java programming course in our department. They have finished the basic Java grammar in the class, and have written simple Java codes in textbooks as exercises. Here, we actually gave them a source code for this assignment that has one error intentionally such that `r + r * 3.14` be corrected to `r * r * Math.PI` for the given radius `r`, and asked them to fix it. We note that these students have used JPLAS to know how to use it before this experiment, and `Math.PI` appears in the test code that is disclosed to the students as shown in Figure 8.

Here, we discuss the background of our experiment. When we introduce new methods, systems, or functions to improve educations in schools, we should verify their effectiveness and clarify problems or disadvantages at their practical use through regular classes. At the same time, the quality of a regular class must be maintained by avoiding disturbances as much as possible. Under this tradeoff, we prepared a full source code having one error that can be corrected within 15min. Actually, we consider that this error correction of an existing source code is appropriate as an assignment to target users of JPLAS, because it can test abilities of students in reading an existing code and debugging it using the functions in JPLAS.

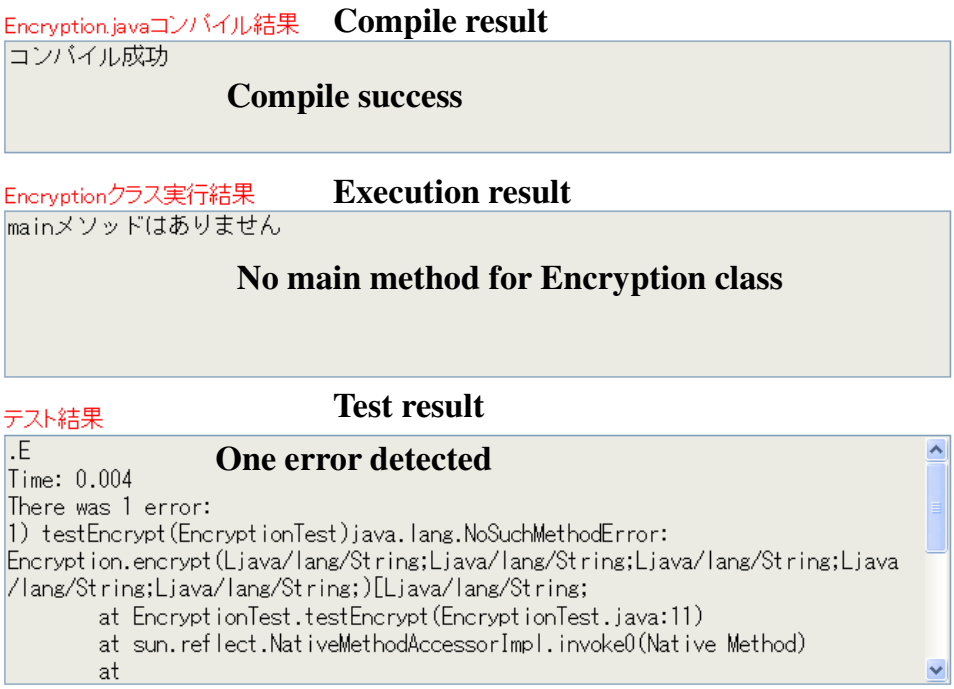

Fig. 6. Test result for source code.

```
JUnit version 4.8.2
.E..E
Time: 0.006
There were 2 failures:
1) testCalCircle(TestMyArea)
java.lang.AssertionError: expected:<113.04> but was:<24.84>
        at org.junit.Assert.fail(Assert.java:91)
        at org.junit.Assert.failNotEquals(Assert.java:645)
        at org.junit.Assert.assertEquals(Assert.java:441)
        at org.junit.Assert.assertEquals(Assert.java:510)
        at TestMyArea.testCalCircle(TestMyArea.java:10)
        at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
            .
            .
            .
```

Fig. 7. JUnit log example.

B. Results from Students

Table I summarizes the distribution of numbers of code submissions by the students for this assignment. Seven students could correctly answer it, where four students repeated submissions twice or more. This result shows that most students could submit answers at least once during this short time, and some students could solve it after correcting codes by referring to outputs from JPLAS.

After the experiment, we asked the students to reply to the six questions in Table II with five grades. Table III shows their replies to the questions.

For Q1, the similar number of students replied positively (4 or 5) or negatively (1 or 2). It indicates that the usability

TABLE I
NUMBERS OF CODE SUBMISSIONS BY STUDENTS.

# of submissions	# of students
0	3
1	18
2	13
3	3
4	2
5	1
6	1
7	0
8	1
Average	1.99

```
1 import static org.junit.Assert.*;
2 import org.junit.Test;
3
4 public class TestMyArea {
5
6         @Test
7         public void testCalCircle(){
8
9                 MyArea tmp = new MyArea();
10                assertEquals( 36*Math.PI,tmp.calCircle(6), 0.000000001 );
11        }
12
13        @Test
14        public void testCalSquare(){
15
16                MyArea tmp = new MyArea();
17                assertEquals( 25,tmp.calSquare(5) );
18        }
19
20        @Test
21        public void testCalArea(){
22
23                MyArea tmp = new MyArea();
24                assertEquals( 36+(36*Math.PI),tmp.calArea(6),0.00000001 );
25        }
26 }
```

Fig. 8. Test code highlighting example.

```
1  public class MyArea {
2          public double calCircle(int r){
3                  return r + r * 3.14;
4          }
5
6          public int calSquare(int length){
7                  return length * length;
8          }
9
10         public double calArea(int fig){
11                 return this.calCircle(fig) + this.calSquare(fig);
12         }
13
14 }
```

Fig. 9. Source code highlighting example.

TABLE II
QUESTIONS FOR QUESTIONNAIRE.

	Question
Q1	Do you think to use the system is easy ?
Q2	Do you think this system is helpful in checking the program functions ?
Q3	Do you think to read the test code is helpful in understanding the assignment specification ?
Q4	Do you feel the response time after the source code submission is long ?
Q5	Do you think the error code highlighting function is helpful in fixing the errors in the code ?
Q6	Do you think this system is helpful in understanding Java programming ?

TABLE III
QUESTIONNAIRE RESULTS WITH FIVE GRADES.

		1	2	3	4	5	
Q1	hard	1	14	13	8	6	easy
Q2	useless	0	9	17	9	7	useful
Q3	useless	0	9	19	10	4	useful
Q4	long	2	5	9	14	12	short
Q5	useless	0	2	22	9	9	useful
Q6	useless	3	5	22	6	6	useful

of JPLAS is basically acceptable for them. However, some students felt that the programming editor on a Web browser *CodePress* is not convenient if compared with an editor on a PC. Thus, a more advanced editor for a Web browser should be introduced in JPLAS.

For Q2 and Q3, most students replied 3 or more, whereas some students replied 2. It indicates that JPLAS is generally helpful in writing a correct Java code for the assignment. For Q2, its negative replies may come from the short time in using JPLAS such that this experiment was over before being accustomed to JPLAS. For Q3, they may come from incomprehension of the TDD method where they did not know how to read the test code. The result for Q6 also indicates the insufficiency. Thus, we should give sufficient instructions of the TDD method before the next experiment, and let students use JPLAS for a longer time there.

For Q4, most students replied positively, whereas some students replied negatively. It indicates that the response

time is generally acceptable, but when many students submit source codes at the same time, the response can be delayed. For Q5, most students replied 3 or more. It indicates that the error code highlighting function is useful in finding errors in source codes. Thus, we expect that this function helps students to study the Java programming by JPLAS more aggressively.

C. Teacher Remarks

Through this experiment, we obtained the following remarks on JPLAS from the teacher teaching the Java programming course:

- The registration of an assignment in JPLAS is easy.
- JPLAS is helpful to improve motivations of students in studying Java programming, because the answering to assignments is easy and outputs of JPLAS are helpful to check the correctness of their answers.
- Because assignments in this experiment are easy, additional experiments with harder assignments are necessary. The selection of proper assignments is important to enhance educational effects for Java programming using JPLAS.
- Handbooks of JPLAS should be prepared so that students can use it without explanations by a teacher.
- JPLAS should provide functions to encourage students to use JPLAS more actively, such as an interface of showing their rankings among the students in the number of correctly solved assignments and/or the total assessing time to JPLAS.

These remarks suggest that this teacher valuated JPLAS positively, and at the same time, it is necessary to implement additional functions, select proper assignments, and prepare handbooks. They will be in our further studies for JPLAS.

VI. Related Works

A number of works have been reported for systems or tools to support programming courses and their applications into classes. Within our surveys, we could not find any Web application system that has the functions of the automatic template generation for a test code to help a teacher and the erroneous line highlighting to help a student in JPLAS.

As *integrated development environments (IDEs)* for programming practices, *Eclipse* [9] and *NetBeans* [10] can plug-in *JUnit*, *Covertura* [8], and *JUnit Helper* [11]. They have a lot of functions to support various needs in developing practical codes by professional programmers, and can be used in programming educations. However, we consider that JPLAS is more suitable for Java programming educations than these existing tools due to the following observations:

- Their affluent functions have been developed to meet various needs of professional programmers in developing practical codes, but not been designed for students in a Java programming course. Such target users of JPLAS usually need considerable time in understanding their proper use, and may feel difficulty in it. As a result, some students may give up using these tools. Besides, preparations of IDEs and related manuals for code testing can be heavy burdens for a teacher who is not familiar to them.
- They assume standalone systems, and need collaborations with other systems for communications between a teacher and students when a teacher presents assignments to students, collects their answers, and promptly feedbacks marking results. As a Web application system, JPLAS can easily realize these essential functions in programming educations. By JPLAS, students can work on Java programming at any place including homes and schools without carrying PCs or software/data, as long as the Internet access service is provided. This portability of study is very important to support Java programming educations in schools.

Here, we note that *JUnit Helper* is a tool to help test code generations, and actually has more functions for use in practical projects than that in JPLAS. In future works, we will consider its incorporation into JPLAS.

In [12], Matsuura et al. presented a lesson support system to improve the programming exercise lesson. This system provides a variety of functions for students and teachers such as uploading and downloading of materials for lessons, registering student records, and creating/evaluating questionnaires. Unfortunately, this system does not support functions for self-studies of students.

In [13], Desai et al. evaluated the effects using the test-first approach (TDD method) versus the test-last approach in early programming courses, and showed that the former one can improve testing and programmer performance, although early programmers are reluctant to adopt it. Thus, to use the TDD method is important to improve the educational quality of programming.

In [14], Desai et al. surveyed the current state of experiments using the TDD method conducted at universities. They show that it can expose students to analytical and comprehension skills needed in software testing, and help them design complex projects and increase confidence.

In [15], Desai et al. demonstrated how the TDD method can be integrated into existing course materials, and showed two controlled experiments where the TDD method was first introduced with unit testing at the beginning of the course, and then, a student needed to write a test code by modifying its similar code. The results indicate that they could successfully develop test codes while learning programming, and the test-first approach using the TDD method gave better results in the quality of developed programs than the conventional non-TDD approach.

In [16][17], Röβling presented a Web-based platform called *WebTasks* for submitting, testing, and discussing student solutions for programming exercises, where student programs are tested by *JUnit*. Unfortunately, this system does not support functions of helping a teacher write a test code, and helping a student find erroneous lines, where the output log of *JUnit* is just displayed.

In [18], Ihantola et al. reviewed recent developments of automatic assessment tools for programming exercises, and discussed their major features and approaches, including programming languages, learning management systems, testing tools, limitations on resubmissions, manual assessments, security, distributions, and specialty.

In [19], Clarke et al. presented an approach of integrating use of software testing tools into programming and software engineering courses. It consists of the development of a Web-

based repository of software testing tools, the training of instructors in testing techniques, and the integration of use of testing tools into programming courses, where results are promising.

In [20], Denny et al. presented and evaluated a Web-based tool providing drill and practice supports for Java programming called *CodeWrite*, where students are responsible for developing exercises that are shared among classmates. Because it does not adopt a testing tool such as *JUnit*, possible variations for program testing are limited.

In [21], Shamsi et al. proposed a graph-based grading system for introductory Java programming courses called *eGrader*. The dynamic analysis of the submitted program is based on *JUnit*, and the static analysis is based on the graph representation of the program. The accuracy was confirmed through experiments.

VII. Conclusion

This paper presented a Web-based *Java Programming Learning Assistant System (JPLAS)* using the *test-driven development (TDD) method*, to enhance educational effects in Java programming courses by assisting self-studies of students while reducing teacher loads. In JPLAS, a source code and a test code can be tested automatically at the server by using an open-source software *JUnit*. Effectiveness of JPLAS with respect to low loads in use for both teachers and students, helps in studying Java programming for students, and the possibility of TDD method educations in Java programming courses, was verified through an experimental application to students in our department.

In future studies, we will improve functions in JPLAS by adopting an advanced programming editor for Web browsers and *JUnit Helper* for test code generations, reduce the response time in testing by improving its codes and allowing multiple servers, and prepare handbooks for teachers and students to use JPLAS. For comprehensive evaluations of JPLAS, we will continue its use in our Java programming course while giving a variety of assignments with different scopes and levels to students, measuring the accessing time and submission times to JPLAS by students, and analyzing the relationships between them and academic scores of the class.

References

[1] JR East (online), http://www.thefreelibrary.com/LEAD\%3A+JR+East\%27s+automated+gate+system+fails+to+read+data+from+Suica...-a0155493841.

[2] ANA (online), http://www.venturedata.org/?i434_Japans-All-Nippon-Airways-computer-system-failure-was-canceled-hundreds-of-flights.

[3] Mizuho Bank (online), http://http://www.reuters.com/article/2011/03/18/mizuho-idUSL3E7EI02L20110318.

[4] K. Beck, *T*est-driven development: by example, Addison-Wesley, Reading, 2002.

[5] JUnit (online), http://www.junit.org/.

[6] N. Funabiki, Y. Fukuyama, Y. Matsushima, T. Nakanishi, and K. Watanabe, "An error code highlighting function in Java programming learning assistant system using test-driven development method," *Lecture Notes in Engineering and Computer Science: Proc. The World Congress on Engineering and Computer Science (WCECS 2012)*, 24-26 October, 2012, San Francisco, USA, pp. 230–235.

[7] CodePress (online), http://sourceforge.net/projects/codepress/.

[8] Code coverage measurement tool *C*obertura (online), http://cobertura.sourceforge.net/.

[9] Eclipse (online), http://www.eclipse.org/.

[10] Eclipse (online), http://netbeans.org/.

[11] JUnit Helper (online), http://code.google.com/p/junithelper/.

[12] S. Matsuura and S. Atsuda, "Lesson support system for improvement of the programming exercise lesson," *Proc. IASTED Int. Conf. Web-based Education*, pp. 131–136, Feb. 2005.

[13] C. Desai and H. Saiedian, "Test-driven learning in early programming courses," *Proc. SIGCSE '08*, pp. 532-536, March 2008.

[14] C. Desai, D. Janzen, and K. Savage, "A survey of evidence for test-driven development in academia," *ACM SIGCSE Bulletin*, Vol. 40, No. 2, pp. 97-101, June 2008.

[15] C. Desai, D. Janzen, and J. Clements, "Implications of integrating test-driven development into CS1/CS2 curricula," *Proc. SIGCSE '09*, pp. 142-152, March 2009.

[16] G. Röβling, "WebTasks: online programming exercises made easy," *Proc. ITiCSE '08*, pp. 363, June 2008.

[17] G. Röβling, "A family of tools for supporting the learning of programming," *algorithms*, vol. 3, pp. 168-182, April 2010.

[18] P. Ihantola, T. Ahoniemi, V. Karavirta, and O. Spepala, "Review of recent systems for automatic assessment of programming assignments," *Proc. Koli Calling '10*, 2010.

[19] P. J. Clarke, T. M. King, E. L. Jones, and P. Natesan, "Using a Web-based repository to integrate testing tools into programming courses," *Proc. SPLASH '10*, pp. 193-200, Oct. 2010.

[20] P. Denny, A. Luxton-Reilly, E. Tempero, and J. Hendrickx, "CodeWrite: supporting student-driven practice of Java," *Proc. SIGCSE '11*, pp. 471-476, 2011.

[21] F. A. Shamsi and A. Elnagar, "An intelligent assessment tool for student's Java submission in introductory programming courses," *J. Intelli. Learning Syst. Appl.*, vol. 4, pp. 59-69, Feb. 2012.

Nobuo Funabiki received the B.S. and Ph.D. degrees in mathematical engineering and information physics from the University of Tokyo, Japan, in 1984 and 1993, respectively. He received the M.S. degree in electrical engineering from Case Western Reserve University, USA, in 1991. From 1984 to 1994, he was with the System Engineering Division, Sumitomo Metal Industries, Ltd., Japan. In 1994, he joined the Department of Information and Computer Sciences at Osaka University, Japan, as an assistant professor, and became an associate professor in 1995. He stayed at University of Illinois, Urbana-Champaign, in 1998, and at University of California, Santa Barbara, in 2000-2001, as a visiting researcher. In 2001, he moved to the Department of Communication Network Engineering (currently, Department of Electrical and Communication Engineering) at Okayama University as a professor. His research interests include computer networks, optimization algorithms, educational technology, Web technology, and network security. He is a member of IEEE, IEICE, and IPSJ.

Yukiko Matsushima received the B.S. degree in information science from Tokushima University, Japan, in 2001, and the M.S. degree in communication network engineering from Okayama University, Japan, in 2010, respectively. In 2001, she joined Be-Max Professional School as a lecturer. She is currently a Ph.D. candidate in Graduate School of Natural Science and Technology at Okayama University, Japan. Her research interests include educational technology and Web service systems. She is a student member of IEICE.

Toru Nakanishi received the M.S. and Ph.D. degrees in information and computer sciences from Osaka University, Japan, in 1995 and 2000, respectively. He joined the Department of Information Technology at Okayama University, Japan, as a research associate in 1998, and moved to the Department of Communication Network Engineering in 2000, where he became an assistant professor in 2003 and an associate professor in 2006, respectively. His research interests include cryptography, information security, and network protocol. He is a member of IEICE and IPSJ.

Kan Watanabe received the B.S., M.S., and Ph.D. degrees in information technology from Tohoku University, Japan, in 2006, 2008, and 2011, respectively. In 2011, he joined the Department of Electrical and Communication Engineering at Okayama University, Japan, as an assistant professor. His research interests include distributed systems and wireless networks. He is a member of IEEE, IEICE, and IPSJ.

Noriki Amano received the B.A. degree from Nihon University, Japan, in 1990, and the M.S. and Ph.D. degrees from Japan Advanced Institute of Science and Technology (JAIST), Japan, in 1996 and 1999 respectively. From 1990 to 1994, he worked at two companies as a system engineer. In 1999, he joined the Graduate School of Information Science at JAIST as an assistant professor. In 2006, he moved to the Center for Faculty Development at Okayama University, Japan, as a senior associate professor. His research interests include educational technology and software engineering. He is a member of IEICE, IPSJ, ACM, and AACE.