Step-by-Step Procedures for answering JPLAS Code Writing Problem

- 1. Firstly, please **extract** the compressed Zip file that we gave with name like "OffJPLAS-task ID", "BASIC", "DATASTRUCTURE" and so on.
- Create a new java project with name "OffJPLAS-task ID" (e.g., OffJPLAS-01) in Eclipse.
- 3. **Copy** all of contents from the extracted folder to src of the created java project.
- 4. Then, please open each package in src and you can see that there are two java classes like classname.java and classnameTest.java for each problem. You must only write the appropriate source codes as you like in classname.java to satisfy the test codes written in classnameTest.java. However, you don't need to write the main function [public static void main (String args[]) in the source code. The example processes can be seen in figure 1, figure 2 and figure 3.



Fig1. Original java class before writing Source code

Thesis - OffJPLAS-01/src/p5/Divide.java - Edipse IDE

Package Explorer 🖄 Ju Junit	 ▶ 11 11 11 + 1 □ 18 ▶ 12 + 19 ▶ 10vide.java 22 DivideTest.java
🛱 Package Explorer 🔀 Jtr JUnit 👘 🗖	🕐 *Divide.java 🕄 🚽 DivideTest.java
Image: Control in the system Library [JavaSE-11] > minite A >	<pre>1 package p5; 2 3 public class Divide{ 4 static int res; 5 static int devide(int n1,int n2) 6 { 7 if(n2>0) 8 { res=(int)(n1/n2); 9 return res; 10 } 11 12 return -999; 13 14 } 15 } 16</pre>
 >	Problems @ Javadoc Declaration Console S3 No consoles to display at this time.

Fig2. Modified java class after writing Source code

File Edit Source Refactor Navigate Search	h Project	Run Target Window Help
📸 🔻 🗐 🔞 🔤 🔹 New Connection	v 5	* 14 🛍 + 🗉 😿 💠 • O + 🎴 - 隆 - 📽 🞯 - 😂 😂
😫 Package Explorer 🖾 🚽 🖉 JUnit 🛛 🗖	Divide	.java 😥 *DivideTest.java 😒
🖻 😫 🗊 🗢	9 pt	blic class DivideTest {
✓ ₩ OffJPLAS-01	11	* 'É¢%`-+, Ìf` fFfbfN
> 🛋 JRE System Library [JavaSE-11]	12	*/
> 🛋 JUnit 4	130	@Test
× ∰ src	14	<pre>public void testi() { int actual = Divide devide(10.5); </pre>
N	16	Assert.assertEquals(10/5, actual);
	17	}
> 010	180	/** * Ž 112 a ž 111
> 🔂 p11	19	* > ₃₃ ,e*Ø,e,c,c*e*+ */
> 🔐 p12	210	@Test
> 🚰 p13	22	public void test2() {
> 💼 p14	23	<pre>int actual = Divide.devide(10,3);</pre>
> 📇 p15	25	Assert.assertEquals(3, actual);
> # p16	260	/**
×	27	* <i>f</i> [<i>f</i> *,Å,ÌŠ,,,èŽZ
	28	*/
> p3	290	/** * f[m Å ÌŠ àŽ7
> 🔂 p4	31	*/
₩ p5	32	
> 🚺 Divide.java	33Θ	@Test
> 🕖 DivideTest.java	34	<pre>public void test3() { int actual = Divide devide(10.0);</pre>
> 🖶 p6	0,36	Assert.assertEquals(-999, actual);
> 进 p7	37	}
170 ET-1	20	

Fig3. Teacher-defined Java class for Test code

5. After this, to solve this code writing problems, you must import JUnit Library. To import the JUnit, right-click on the [Off JPLAS-task ID] and click[Build Path] => [Add Libraries] => [JUnit4] => Finish. The example can be seen in figure 4.



Fig.4 Importing JUnit Library

6. Then, students can check by themselves whether or not the source codes satisfy the test codes by right-click on the classnameTest.java and then go to [Run as] => [JUnit Test]. JUnit will show your test result in eclipse IDE. "Green bar" means your source code is correct and "Brown bar" means your code includes failure or error. You can find errors in failure trace. This processes can be seen in figure 5 and figure 6.

Thesis - Off JPL		New	>	1
		Open Open With	F3	₩ • 0 • % • % • # G • # Ø Ø
12 Package Explorer		Open Type Hierarchy	F4	d×8
V ES OFFICAS-01		Show In Show in Local Terminal	Alt+Shift+W >	
> mil. JRE System > mil. Junit 4	10	Copy Copy Qualified Name	Ctri+C	e(int n1,int n2)
> 🔂 p1	×	Paste Delete	Ctrl+V Delete)(n1/n2); =;;
> # p11	.80	Remove from Context	Ctrl+Alt+Shift+Down	1
> 🔂 p12		Source	Alt+Shift+S >	
> 🔠 p14		Refactor	Alt+Shift+T >	
> # p15	5.8	Export		
> # p3		References Declarations	>	
> in ps	s	Refresh Assign Working Sets	FS	1
> 10 Divis	9	Coverage As	>	
> 进 p7	0	Run As	>	Jig 1 JUnit Test Alt+Shift+X, T
8q 🚓 <	粋	Debug As	*	Run Configurations
> 👘 p9 > 🕀 report		Restore from Local History	3	Saration Console 23
i pmd-java		Team	3	
QuestionElank		Compare with	2	
<		Validate		
p5.DivideTest.java - C		Properties	Alt+Enter	

Fig.5 Testing with JUnit



Fig.6 Comparing Testing Results from JUnit

- 7. Students **can repeat** the verification of test code with JUnit every time they modify the source code, until the test is successful.
- 8. After solving all of problems from each package, to leave a test log ,please right-click on this current java project and go to "Export" => "Ant Buildfiles" >> "Next" >> select the project folder OffJPLAS-task ID (e.g., OffJPLAS-01) and click Finish. This can be seen in figure 7 and figure 8.

File Edit So	New	> Indow Help	Export –
	Go Into		Select
Package Dx	Open in New Window Open Type Hierarchy Show In Alt+Shit	F4 ft+W> ft+W>	Generates Ant Buildfiles for Eclipse Java projects.
> 20 XH > 20 XH > 20 XH > 20 XH > 20 XH > 20 XH	Copy CC Copy Qualified Name Paste C	bi+C test1() { al = Divide.devic ssertEquals(10/5,	Select an export wizard:
¥3冊(49日 3冊(3冊(3冊(3冊(Delete D Riemove from Context Coti+Alt+Shift+E Build Path Source Alt+Shi Refactor Alt+Shi	wete Down test2() { al = Divide.devic eft+T> csentEquals(3, a)	↓ Archive File ↓ File System ↓ Preferences > >> >> C/C++
· 문· · 문· · 문·	Import Export	",ėžz	> 🎦 Install
令 3 晤 (3 晤 (3 晤 (3 晤 (Refresh Close Project Close Unrelated Projects Assign Working Sets	F5 wyłżz	> 😕 Run/Debug > 🔁 System Management > 🔁 Tasks
) ## c) ## c) ## c) ## c) ## c) # pmd-ja.	Coverage As Run As Debug As Profile As Restore from Local History	<pre>> a1 = Divide.devic scartEquals(+999, > > > Declaration</pre>	> 👝 теат Х 👝 ума
Test Test2C Fest2C Fest2C Fest2C	Team Compare With Configure Validate	Ling C: Program Hies Li	(?) <back next=""> Finish Cancel</back>
HTPLAS-01	Properties Alt+6	Enter	

Fig.7 Steps for Generating Build file

enerate Ant Buildf	iles	the
Generates Ant buildfiles	-16-	
Select the projects to us	e to generate the Ant buildfiles:	
	Problem	Select All
	S1	Decelect All
C COFFJPAS-PHD-3		Deselect All
C COFFJPLAS_4		
Thesis		
🔲 😂 java-zip-starter	-project-master	
C COFFJPLAS-01		
☐ Check projects for Ar ☐ Create target to com	nt compatibility pile project using Eclipse compiler	
Name for Ant buildfile:	build.xml	
Name for Ant buildfile:	build.xml	
Name for Ant buildfile: JUnit output directory:	build.xml junit	

Fig.8 Last Step for Generating Build file

9. Also, **create a folder** named "junit" by yourself under "OffJPLAS-task ID" as shown in following figure 9.



Fig.9 Folder Creation

- 10. Then, select the resulted "build.xml" under the project and right-click on "build.xml" => "Run as" => "2. Ant build". In the "Edit configuration and launch" box, select three types of targets such as "build [default]", "all test classes to execute (e.g., DivideTest, , MaxItemTest,...)", and "junit report". Then, click "apply" and "run". The XML file of the test result of each problem will be outputted to the "junit" folder.
- 11. After that, you will see the "**BUILD SUCCESSFUL**" information in the console as in the following figure 10.



Fig.10 Generation of XML file of the test result of each problem

* Correspondence when it becomes BUILD FAILED, add tool.jar

[Window] - [Preference] - [Ant] - [Runtime] - [Global Entries] [Add External JARs] - [C:\Program Files\Java\jdk-****\lib\tool.jar]

12. Then, select report.java in the report folder under src and run it. When you execute "report.java" of the assignment project, enter the student number (8 digits) in the submission input window and click the "OK" button (see in figure 11).



Fig.11 Submission Input Window

13. Then, the "Save" screen will be displayed. Enter the student number again, press the "Save" button and then the answer data is saved as the Zip folder "xxxxxxx.zip" in your computer (see in figure 12).

Save In: 😂	_ocal Disk (D:)	- A	
C_offline_	/alue_trace_Revise9.zip		
PPLASEx	cels.zip		
PPLAS Re	sults.zip		
	ID-12345		
File <u>N</u> ame:			
File <u>N</u> ame: Files of <u>T</u> ype:	zip		
ile <u>N</u> ame: iles of <u>T</u> ype:	zip		

Fig.12 Save Window

14. Submit the answer file "xxxxxxx.zip" to the teacher using a USB memory or email. Finally, you have successfully solved code writing problems.