北星学園大学経済学部北星論集第52巻第２号（通巻第63号）（2013年３月）・抜刷

汎用模型化言語（UML）が組込まれたJava応用プログラム開発プラットフォーム（その１）

能　登　　宏

A Java Application Development Platform with a Unified Modeling Language (UML) Plug-in (Part I)

Hiroshi NOTO

Contents

1. Introduction

In this article an Object-Oriented Java Programming Environment[1] is introduced incorporating Unified Modeling Language[2] (UML) as a platform on the computer system.

We have constructed a Java programming platform for the students to learn and pursue application development, of the five courses that the author is in charge of, Department of Management and Information of Hokusei Gakuen University. We have been thinking and planning these two or three years that the programming platform in our courses that is built on the client computers and servers in the PC rooms of the Information Systems Center in our university is to meet those requirements:

- to support the Integrated Environment based on Object-Oriented Programming
- to manipulate Domain Modeling[3] prescription
- to incorporate Unified Modeling Language (UML)
- to facilitate using a database system
- to enable us to design reporting layouts

Those requirements, of course, reflect the recent progress in software engineering and software development methodologies. The setup of the learning and developing platform satisfying the above requirements makes the students to concentrate on Domain Modeling

キーワード：Java Application, Database Management System, NetBeans IDE, Unified Modeling Language(UML), Reporting System

with Unified Modeling Language using the database system without specifying a particular programming language. In the stage targeting the implementation, however, of our domain modeled deliverable we have to specify a particular programming language.

The purpose of our seminars and lectures is for the students to study and pursue their software development practice based on those programming methodologies without taking much time or effort in learning the grammar in depth of the programming language or in getting used to the platform software. We hope that the students are able to manipulate their own logic visually in the domain modeling by trial and error on the introduced platform to decide their business models and architecture specifications which are to result in their own applications.

In §2 the basic design of our platform complex is presented. Explained in §3 is each configuration of several combinations of our platform components in our courses. We set forth in §4 the case studies of actual practices of running the platform in our courses. In §5 some feedback of our platform is noticed. The conclusion of the present article is stated in §6.

2. Basic Design of Platform Complex

In our seminars and lectures we have adopted Java as the programming language and NetBeans[4] as the integrated development environment (IDE) in Java. We have decided to take advantage of NetBeans features to help us develop our Java applications efficiently. In addition to the standard support of Java application development, NetBeans leverages IDE either by built-in components or by installing additional plug-ins. NetBeans also supports many APIs (Application Programming Interfaces) to facilitate NetBeans functions and standalone applications.

The basic design of our platform is a combination of NetBeans and additional plug-ins that meet our demands for our Java programming and software development environment which are described in the previous section. We have realized the following combination of NetBeans and plug-ins: NetBeans for Object-Oriented Programming plus Unified Modeling Language for domain modeling technique and software development architecture. We have implemented Visual Paradigm's SDE, i.e. SDE EE NB[5] as the UML tool which comprises a UML design tool and a UML CASE (Computer Aided Software Engineering) tool. In addition to that we further need a database system and a reporting or documentation system. In the present case we have introduced MySQL[6] as the former (a database system) and iReport[7] as the latter (a reporting or documentation system). NetBeans generates JPA entities from an existing database schema. Here JPA is Java Persistence API, the standard Object-Relational Mapping tool included with Java EE[8].

In practice one combination of NetBeans and plug-ins is symbolically written as follows:

NetBeans + MySQL + iReport + UML (SDE EE NB)

Those components synthesize our platform complex that is oriented towards the domain modeling and visually designing report layouts, in our courses for software application development.

Another combination of NetBeans and additional plug-ins in our courses is that for the Web application development. The web application is another important field or subject of software development studies and practices. Our setup of the web application platform is like this: the basic setup mentioned above incorporates web oriented plug-ins: Web Container such as GlassFish application server[9], Java Frameworks which comprise JSF (Java Server Faces) and/or ICEfaces[10]. The NetBeans web application basically makes web application development efficient by using Servlet APIs and Java Server Pages (JSP). In addition to web development, NetBeans allows us to easily develop Enterprise JavaBeans (EJBs). An EJB is a server-side, reusable component (introduced as part of the Java EE 6 specification) that encapsulates specific business logic and is activated and executed by the EJB container within an application server.

As a result another combination of NetBeans and plug-ins for Web application development is symbolically written as follows:

NetBeans + Web Server + Web Container + Java Framework (s) + MySQL + iReport + UML (SDE EE NB).

3. Configuration of Combinations of Platform Components

In this section the configuration of each combination of NetBeans and plug-ins is elaborated. Next we briefly explain the NetBeans Platform architecture. We describe, however, the conceptual structure of NetBeans IDE first. It should be noted that Java Development Kit (JDK)[11] is a prerequisite installation for NetBeans.

The NetBeans IDE is an open-source Integrated Development Environment that is used throughout our courses and makes it easier for us to develop and deploy our applications. The functionality and characteristics of an IDE are created in the form of modules on top of the NetBeans Platform which is explained in the next paragraph. The base IDE includes those functionalities, such as an advanced multi-language editor, debugger and profiler integration, file versioning control and unique developer collaboration features in addition to Navigator API and Refactoring API and other APIs. The NetBeans profiler can provide important information about the runtime behavior of our application; monitors thread states, CPU performance, and memory usage of our application from within the IDE, and imposes relatively low overhead. In NetBeans “refactoring” is a disciplined technique for improving the structure of existing code without changing the observable behavior, e.g. rename, replace block of code with a method and so on. NetBeans IDE is a very good example of a modular-rich client application. The conceptual structure of the NetBeans IDE is displayed in Fig. 3-1 cited from 12).

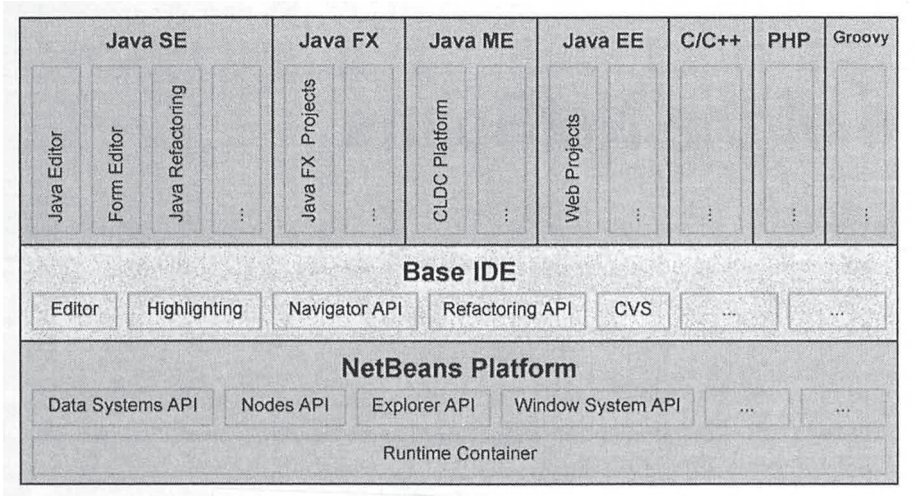

Fig. 3-1. Conceptual structure of the NetBeans IDE

The NetBeans Platform is a broad Java framework on which we can base large desktop applications. The basic building block of the NetBeans Platform is modules. A module is a collection of functionally-related classes together with a description of the interfaces that the module exposes. The interface is provided by the Windows System API through the TopComponent class group for multiple windows. The complete NetBeans Platform, as well as the application built on top of it, is divided into modules. These are loaded by the core of the NetBeans Platform which is known as the NetBeans "runtime container". The runtime container loads the application's modules dynamically and automatically. The runtime container is also responsible for running the application. To optimize the encapsulation of code within modules, which is necessary within a modular system, the NetBeans Platform provides its own "classloader system" which is a part of the NetBeans runtime container. Each module is loaded by its classloader. To use the functionality from other modules, a module can declare dependencies on other modules. These dependencies are declared in the module's manifest file and resolved by the NetBeans runtime container, ensuring that the application always starts up consistently. As shown in Fig. 3-2 cited from 12), the NetBeans Platform itself is formed from a group of core modules which are needed for starting the application and for defining its user interface. To this end the NetBeans Platform makes many API (Application Program Interface) modules and service provider interface (SPI) modules available, simplifying development processes considerably.

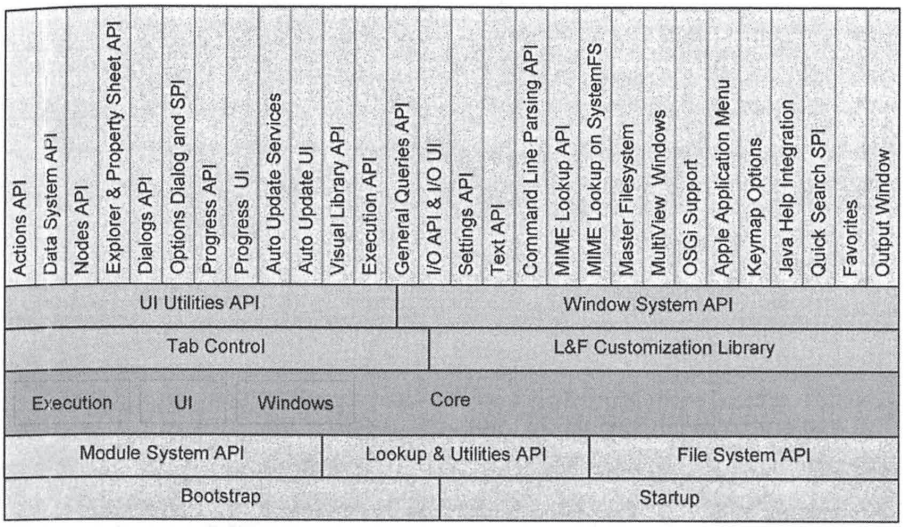

Fig. 3-2. NetBeans Platform Architecture

NetBeans creates many types of projects. We can develop applications based on those types of projects. The NetBeans Platform application project type is one of the project types built on the NetBeans Platform. In this project only the NetBeans Platform modules are provided by default. However we may have the possibility of accessing any modules of the NetBeans IDE.

Finally we mention the application server in our case. NetBeans needs to be connected to GlassFish server so that applications built with IDE can be easily deployed to the application server. As a result we have our Web application deployed and make it available on the application server.

3.1 Combination of NetBeans and UML tool

In this combination, the Java application gets the capability of visual modeling for the object. An object is a self-contained entity with well-defined characteristics and behaviors while the characteristics and the behaviors are represented by attributes and operations respectively. A class is the generic definition for a set of similar objects and an object is an instance of the class. An object model provides a static conceptual view of an application. It shows the key components (objects) and their relationships (associations) within the application system. In our Visual Paradigm (VP) UML tool, more specifically the Smart Development Environment Enterprise Edition (SDE EE) in the present case, a class diagram can be used to draw the objects and classes inside a system and the relationships between them.

A visual modeling for the object model brings about not only creating a new object model, but transforming from a data model. A data model provides the lower-level detail (or entity) of a relational database of an application. It shows the physical database models and their relationships in the application. An Entity Relationship Diagram can be used to

describe the entities inside the system and their relationships with each other. As Object-Relational Mapping (ORM) is automated, the database, code and persistent layer can be generated, which in turn makes streamlined the model-code-deploy software development process.

SDE is not only a visual UML modeling plug-in, but an Object-Relational Mapping (ORM) tool: SDE automates the mappings between Java objects, object models, data model and relational database. SDE supports not only the generation of persistent code and database, but the synchronization between persistent code, object model, data model and relational database, which yields a significant reduction of the development time.

In Visual Paradigm UML the Object Model (UML diagram, Class Diagram in particular) generates the Java persistent code and the Object Model maps to the Data Model (i.e. database creation, property setup and data assignment). The persistent code is the object that enables to store and retrieve data in relational databases permanently. This function, eventually supports our software development in database applications in an easier way.

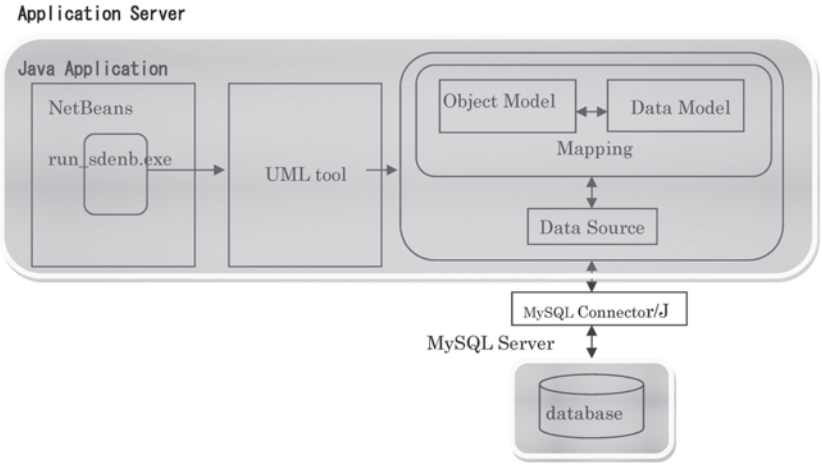

Fig. 3-3. Combination of NetBeans and UML tool

MySQL is a popular open source Relational Database Management System (RDBMS) commonly used in web applications due to its speed, flexibility and reliability. MySQL employs SQL, or Structured Query Language, for accessing and processing data contained in databases. MySQL Connector/J is an implementation of Sun's JDBC 3.0 API for the MySQL relational database server. It strives to conform as much as possible to the JDBC API. The Java Database Connectivity (JDBC) API is the industry standard for database-independent connectivity between the Java programming language and a wide range of databases. MySQL Connector/J is known to work with Application Servers, Object Relational Mapping Tools, Development Environments like NetBeans.

3.2 Combination of NetBeans and iReport

The combination of these software components is for creating reports in NetBeans applications. To this end the iReport plug-in needs to be implemented. In the iReport plug-in, the viewer program is required to use the JasperReort API in the NetBeans project. We also need to add the MySQL JDBC driver to our library in the NetBeans project, since iReport accesses a database to show data. The Java Database Connectivity (JDBC) technology is used to access a database from NetBeans. The iReport Designer facilitates designing a report (report layout) and printing it a great deal.

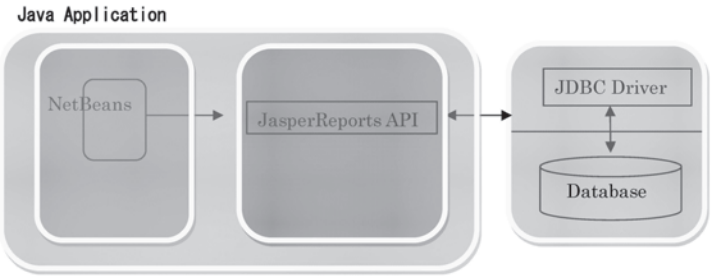

Fig. 3-4. Combination of NetBeans and iReport

3.3 Combination of NetBeans and MySQL on GlassFish Server

The combination of this setup is for a simple web application that connects to a MySQL database server. MySQL employs SQL (Structured Query Language) for accessing and processing the data contained in the database. The most efficient way to implement communication between the server and the database is to set up a database connection pool. Creating a new connection for each client request can be very time-consuming. To rectify the situation, numerous connections are created and maintained in a connection pool. Any incoming requests that require access to the application's data source use an already-created connection from the pool. Likewise, when a request is completed, the connection is not closed down, but returned to the pool.

The GlassFish Server provides a lightweight modular server for the development of Java Platform Enterprise Edition (Java EE) 5 or 6 applications and Java Web Services. GlassFish, therefore, is the reference implementation of Java EE Application Server. An application server is a piece of software that serves applications through the Internet to provide a service. Java EE Application Servers do this by implementing the Java EE Specification[8]. GlassFish is characterized by its enterprise performance, scalability, and reliability. The main deliverables of GlassFish are an Application Server, the Java EE 5 or 6 Reference Implementation, and the Java Persistence API Reference Implementation.

As for configuring JDBC Connection Pools, a connection pool contains a group of JDBC connections that are created when the connection pool is registered, i.e. when starting up a GlassFish Server or when deploying the connection pool to the target server or the cluster. Connection pools use a JDBC driver to create physical database connections. Our

application borrows a connection from the pool, uses it, and returns it to the pool when closing it.

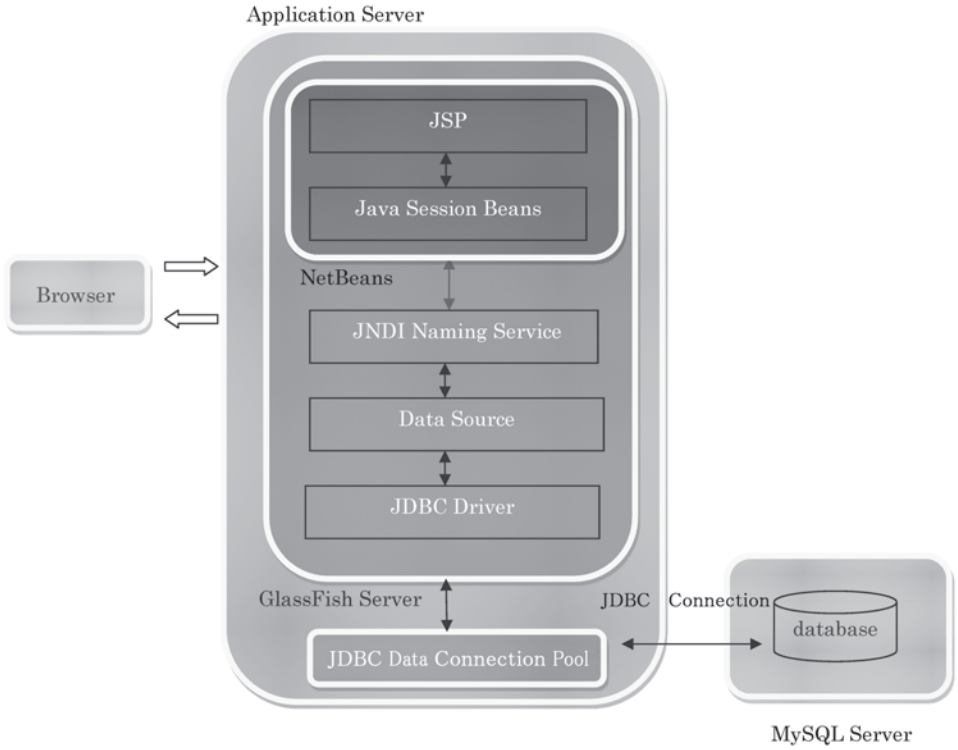

Fig. 3-5. Combination of NetBeans and MySQL on the GlassFish Server

3.4 Combination of NetBeans Platform and Java DB

The Java DB database system is implemented entirely in Java and Java DB is delivered as the client database by default in Java Platform 6 and 7. The NetBeans IDE supports Java DB. The actual database system is embedded as the file derby.jar and it also makes its driver available. A Java DB database can be integrated into a NetBeans Platform application. We create entity classes from JavaDB and wrap the entity classes into a module together with modules for the related Java Persistent API interface. Thereby we acquire a means to have the code for accessing the database.

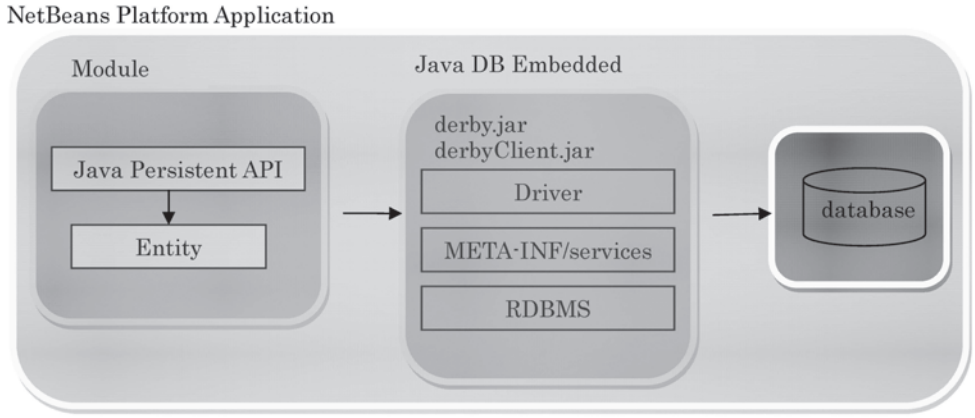

Fig. 3-6. Combination of NetBeans Platform and Java DB

4. Case Studies of Running the Platform

In this section we present several Java application deliverables which have been developed based on our Java platforms with specific combinations of NetBeans plus plug-ins in our seminars and lectures in 2012. Listed below in each case study is a type of combination rather symbolically followed by the name of our deliverable in the course. After that the Java application developed in the course is briefly explained and finally some screen shots are displayed that show what the application looks like and significant and impressive scenes brought about with the help of the plug-ins, implemented on our platform.

4.1 NetBeans + UML

【Seminar I】
"Ticket Reservation System"

This application deals with the basic operations of making reservations of tickets for some events, matches, concerts, books and movies and the like. The basic operations here are *Show the Ticket Information*, *Reserve Tickets*, *Show and/or Confirm Reservations*, *Change Reservations*, *Cancel Reservations* and *Exit*.

In the process of application development the class diagrams are constructed of the objects pertaining to the present domain of *Reserving Tickets* with the help of the UML tool VP SDE. Figure 4-1 shows an example of the class diagrams drawn for the package "Application" that comprises four classes: *Ticket*, *TicketCatalogue*, *Reservation* and *ReserveCatalogue*.

As mentioned in the previous section, Smart Development Environment (SDE) is not only a visual UML modeling tool, but an Object-Relational Mapping plug-in with IDE which supports building a database application and automates the mapping between object model

and data model. This means that SDE is capable of generating Java code from data model and object model. We demonstrate here the code generation from the class diagram (see Fig. 4-1) of the package "Application" of the present application in Fig. 4-2 and List 4-1 where the generated code of the class diagram "Ticket" is displayed.

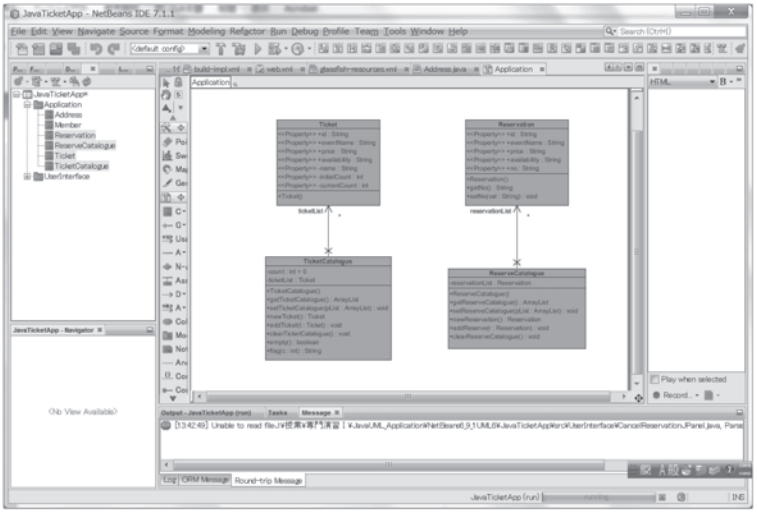

Fig. 4-1. Class diagram of the package "Application" of the present application drawn with SDE

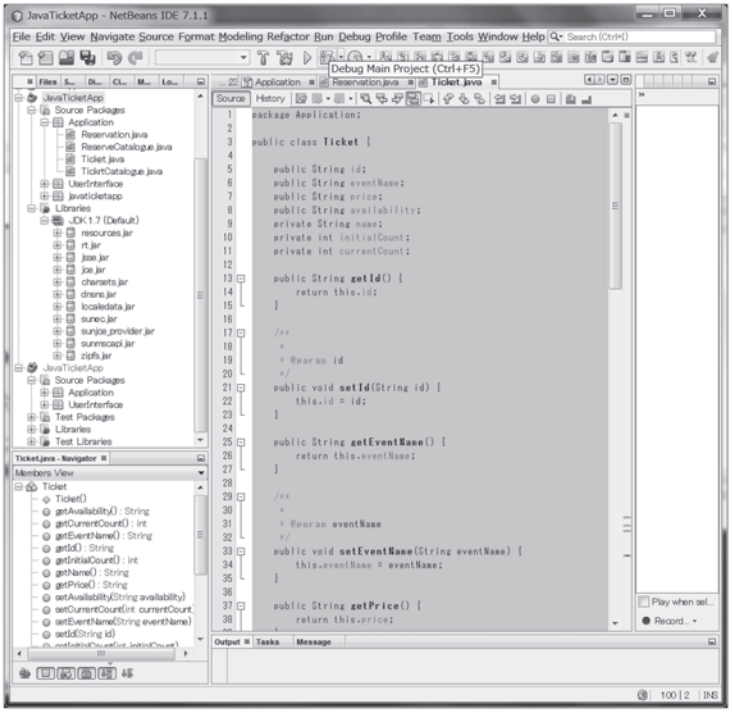

Fig. 4-2. Screen shot of the generated code on NetBeans from the class diagram "Ticket" on SDE

```
package Application;

public class Ticket {

	public String id;
	public String eventName;
	public String price;
	public String availability;
	private String name;
	private int initialCount;
	private int currentCount;

	public String getId() {
		return this.id;

	}

	/**
	 *
	 * @param id
	 */
	public void setId(String id) {
		this.id = id;
	}

	public String getEventName() {
		return this.eventName;
	}

	/**
	 *
	 * @param eventName
	 */
	public void setEventName(String eventName) {
		this.eventName = eventName;
	}

	public String getPrice() {
		return this.price;
	}

	/**
	 *
	 * @param price
	 */
	public void setPrice(String price) {
		this.price = price;
	}

	public String getAvailability() {
		return this.availability;
	}

	/**
	 *
	 * @param availability
	 */
	public void setAvailability(String availability) {
		this.availability = availability;
	}

	public String getName() {
		return this.name;
	}

	/**
	 *
	 * @param name
	 */
	public void setName(String name) {
		this.name = name;
	}

	public int getInitialCount() {
		return this.initialCount;
	}

	/**
	 *
	 * @param initialCount

	 */
	public void setInitialCount(int initialCount) {
		this.initialCount = initialCount;
	}

	public int getCurrentCount() {
		return this.currentCount;	}
	/**
	 *
	 * @param currentCount
	 */
	public void setCurrentCount(int currentCount) {
		this.currentCount = currentCount;
	}
	public Ticket() {
		throw new UnsupportedOperationException();
	}
}
```

List. 4-1. Generated code in the text format from the class diagram *Ticket* on SDE

Fig. 4.3 displays the initial menu screen where a button in the left pane starts up its operation shown in each button's text. From Fig. 4–4 to Fig. 4–10 we demonstrate how a screen switches to another according as we click on a button from *ViewTicketInfor* to *CancelReservation.*

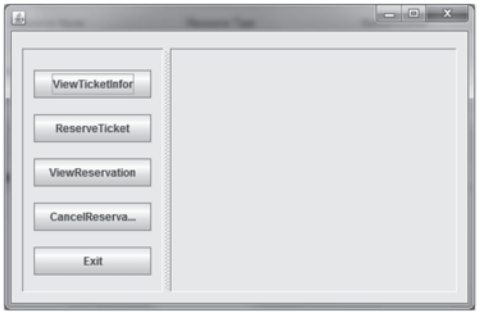

Fig. 4–3. Initial menu screen

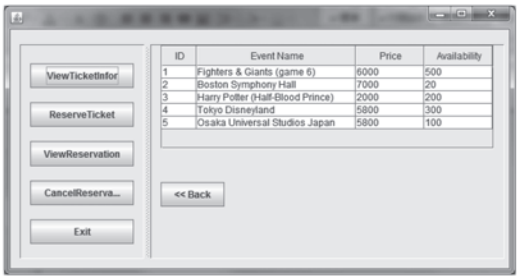

Fig. 4–4. Viewing the "Ticket Information" dealt with in the application

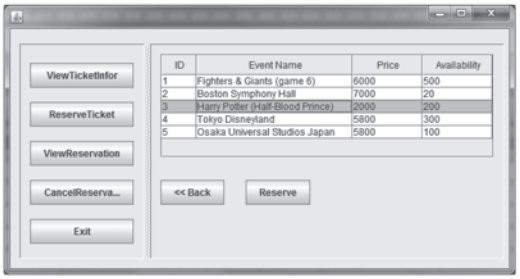

Fig. 4–5. Making reservations for ticket #3

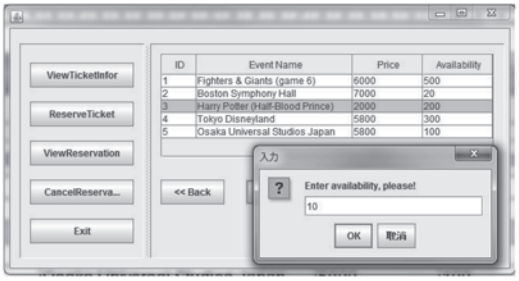

Fig. 4–6. Specifying 10 tickets to be reserved

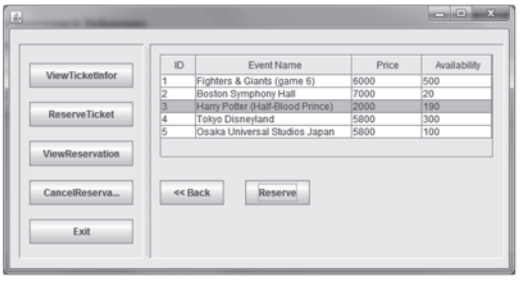

Fig. 4–7. Processing a reservation which results in the update of availability of ticket #3

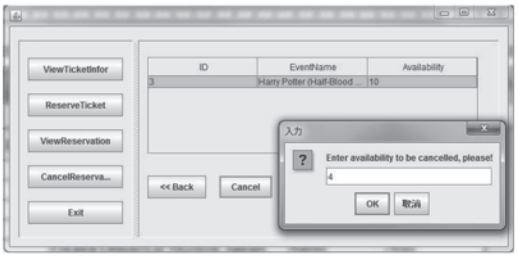

Fig. 4–8. Cancelling 4 tickets for ticket #3

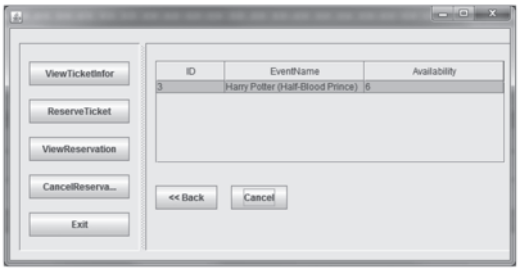

Fig. 4-9. Processing a cancellation which yields 6 tickets now reserved

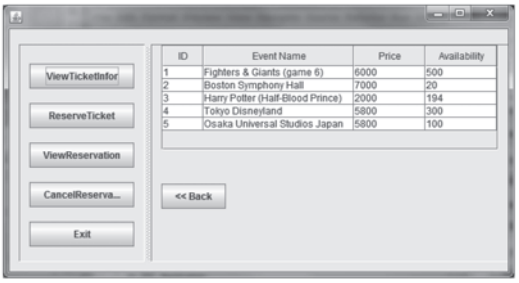

Fig. 4-10. Viewing the present Ticket Information available in the application

4.2 NetBeans + UML + JSP + GlassFish + MySQL

【Application II】
"HirosiIFPWAFCAD System"

We owe this application mostly to a tutorial presented by NetBeans developers[13]. The application creates a simple web application that connects to a MySQL database server. The application we build involves the creation of two JSP (Java Server Pages) pages. In each of them we use HTML and CSS (Cascading Style Sheet) to implement a simple interface, and apply JSTL (JavaServer Pages Standard Tag Library) technology[14] to perform the logic that directly queries the database and inserts the retrieved data into the two pages. The two database tables *Subject* and *Counselor* are contained in the MySQL database "mynewdatabase."

We have setup a platform based on the NetBeans IDE incorporated by the VP UML tool (SDE EE). The NetBeans platform is connected to the MySQL database and contained in the Application Server Java EE with the GlassFish server plugged-in. In our web application development, the Java Server Page (JSP) technology is used. JSP contains HTML tags and Java code as well. As long as the Java code is contained in a web application, JSP will be able to process the form data that is sent to it. The configuration of the combination of the present plug-ins requires JavaServer Pages Standard Tag Library (JSTL), the Java Database Connectivity (JDBC) API, and two-tier client-server architecture.

We have constructed two tables *Subject* and *Counselor* in the MySQL database named "mynewdatabase." As stated in Section 3, the GlassFish Server provides the connection pooling functionality. In order to take advantage of this functionality, we have setup a connection pool named "HirosiIfpwafcadPool" and configured a JDBC (Java Database Connectivity) data source "jdbc/hirosiIFPWAFCAD" for the server which our application can use for connection pooling.

We show the result of running the present application in the browser. When the first JSP file (*index.jsp*) appears in the browser, we select a subject from the drop-down list

and click the submit button (see Fig.4-11). Then we should now be forwarded to the second JSP file (*response.jsp*) page that shows details corresponding to our selection (see Fig. 4-12).

Fig. 4-11. Displaying the content in the database table *Subject* and a subject *Crisis Management* selected from the drop-down list box.

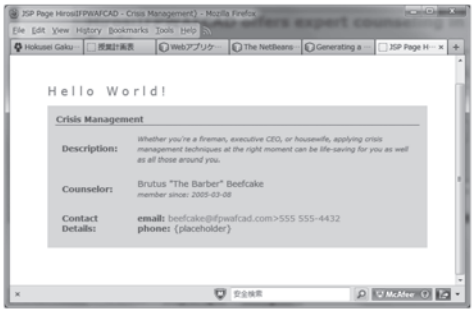

Fig. 4-12. Showing the detailed information from the database table *Counselor* corresponding to the selected subject *Crisis Management*

4.3 NetBeans + UML + MySQL + iReport

【Software I and II】
"Arbeit Wage Calculation System"

This application calculates the wages for part-time workers (Arbeit [or Arubaito in Japanese] and prints the invoices for their wages. The basic operations of the system are represented by the buttons' texts in the left pain of the menu screen： *Select Arbeit Code*, *Input Arbeit's Work Hours*, *Display Arbeit's Wage Invoice*, *Print Arbeit's Wage Invoice* and *Exit* (see Fig. 4-13).

As explained in Section 3, Smart Development Environment (SDE) is an Object-Relational Mapping (ORM) tool as well. SDE, therefore, supports ORM for Java (called Java ORM) and we can reverse engineer the Java classes into the object model with *ORM-Persistable* stereotyped. The *ORM-Persistable* class is capable of manipulating the persistent data with the relational database. This means that the Java classes written in the Java

code are reverse engineered into the object model, i.e. their corresponding class diagrams. Furthermore the ORM diagram provides a view of mapping between a *ORM-Persistable* class and its entity (called Class Mapping).

In the process of development of the application we reverse engineer the package "UserInterface" which contains the main frame of the Initial Menu Screen into the object model in order to view and confirm the object structure of our application visually. Shown in Fig. 4-13 is the Initial Menu Screen of the system which appears when we invoke the application. The user interface of the system is coded in Java and stored in the package "UserInterface."

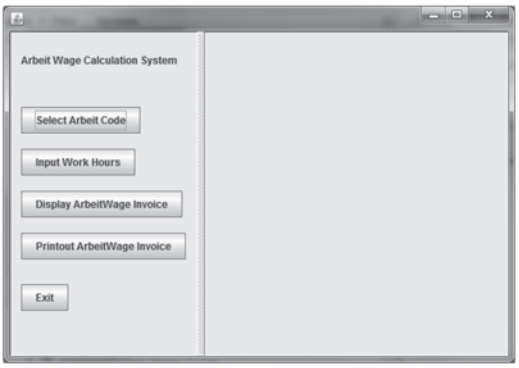

Fig. 4-13. Displaying the Initial Menu Screen of the "Arbeit Wage Calculation System"

In Fig. 4-14-1 we reverse engineer the "UserInterface" package into the object model which displays the class structure of the package visually with the capability of the ORM mapping in SDE implemented on our NetBeans platform. It should be noted here that *stereotype* 《ORM Persistable》 is assigned above the class name in each class diagram. The reverse engineered class diagrams in the package "UserInterface" are *MainJFrame, SelectArbeitCodeJPanel*, *InputHoursAndDaysWorkedJPanel*, *DisplayArbeitInvoiceJPanel* and *PrintArbeitInvoiceJPanel* which correspond to their respective buttons except *MainJFrame* in the Initial Menu Screen. The *MainJFrame* class corresponds to the Initial Menu Screen itself. Furthermore the mapping between the object model and the data model then brings about the mapping between the *ORM-Persistable* class and its corresponding entity. In Fig. 4-14-2 the class diagrams in the package "UserInterface" are *synchronized* to their Entity Relationship Diagram (ERD).

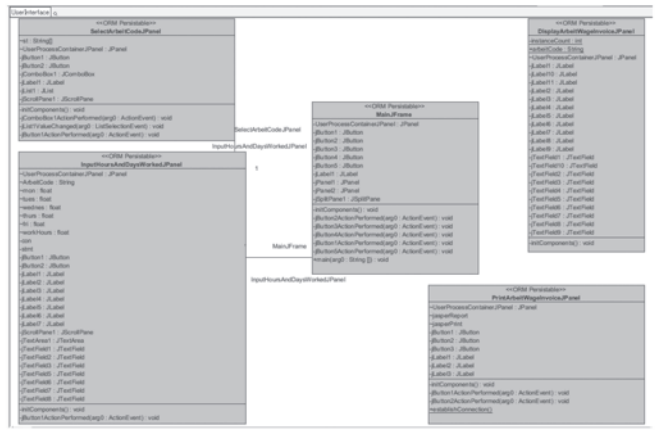

Fig. 4–14–1. Class diagrams in the package “UserInterface” reverse engineered from the corresponding Java objects

Fig. 4–14–2. Entity Relationship Diagram （ERD） *synchronized* from the class diagrams in the package “UserInterface”

From Fig. 4–15 to Fig. 4–18 we demonstrate how one screen switches to another according as we click on a button from *SelectArbeitCode* to *PrintoutArbeitWageInvoice.*

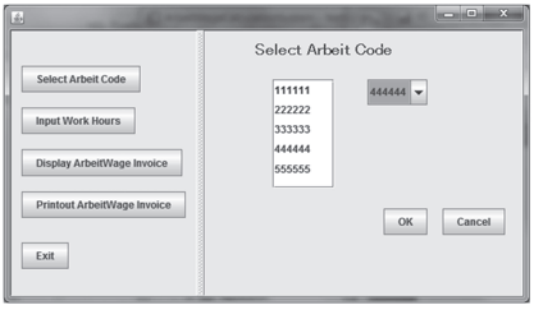

Fig. 4–15. Clicking on the *OK* button after selecting Arbeit Code “444444”

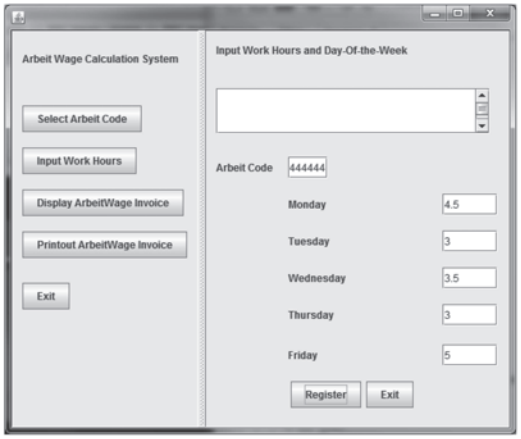

Fig. 4-16. Inputting work hours from Monday to Friday for the selected Arbeit

Fig. 4-17. Displaying the Arbeit Wage Invoice for the selected Arbeit

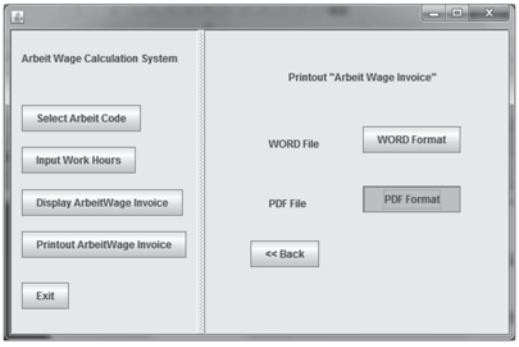

Fig. 4-18. Printing the Arbeit Wage Invoice in the PDF format

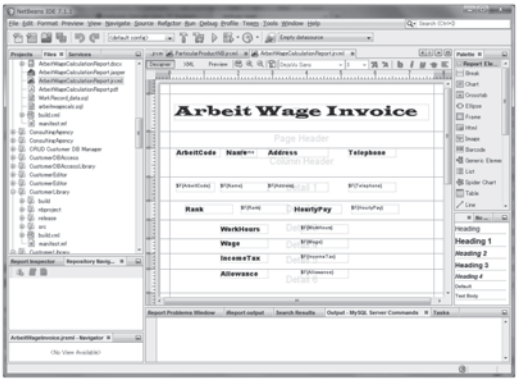

Fig. 4-19. Designing the layout of the Arbeit Wage Invoice in the iReport Designer within the NetBeans work area

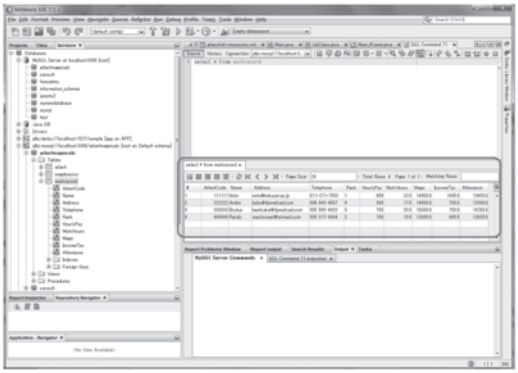

Fig. 4-20. The contents of the Arbeit Wage Invoice displayed in Fig. 4-17 are dynamically populated in the *workrecord* table in the "arbeitwagecalc" databese.

Figure 4-19 depicts iReport plug-in's capability of a visual report designer integrated in the NetBeans platform. Here the "Arbeit Wage Invoice" layout is designed through the main user interface of iReport Designer within the NetBeans work area.

In the "Arbeit Wage Calculation System", the contents of "Arbeit Wage Invoice" displayed in Fig. 4-17 are dynamically populated in the *workrecord* table （see the enclosed area by a rectangular in Fig. 4-20） contained in the MySQL database "arbeitwagecalc" that has already been created in advance and that we have registered a connection for in the NetBeans IDE.

The "Arbeit Wage Invoice" for the part-time workers is printed in the PDF format in Fig. 4-21-1 and Fig. 4-21-2.

Fig. 4-21-1. Arbeit Wage Invoice for the part-time workers printed in PDF format

Fig. 4-21-2. Arbeit Wage Invoice for the part-time workers printed in PDF format (continued)

References

1) Mary Campione and KathyWalrath. The Java Tutorial Second Edition: Object-Oriented Programming for the Internet, Sun Microsystems, Addison Wesley, 1998.
Cay S. Horstmann, Gary Cornell. Core Java Volume I and II, eighth edition, Prentice Hall, 2008.
2) Tom Pender. UML Bible, Wiley Publishing Inc., 2003.
3) Tim Howard. The Smalltalk Developer's Guide to VisualWorks, Cambridge University Press, 1998.
http://c2.com/cgi/wiki?DomainModel: A DomainModel is an object model of a problem domain. Elements of a domain model are DomainObject classes, and the relationships between them.
4) http://netbeans.org/
5) http://www.visual-paradigm.com/
6) http://dev.mysql.com/
7) http://community.jaspersoft.com/project/ireport-designer
8) http://docs.oracle.com/javaee/
9) http://glassfish.java.net/
10) http://www.icesoft.org/java/
11) http://www.java.com/en/download/faq/develop.xml
12) Heiko Boeck. The Definitive Guide to NetBeans Plaform 7, Apress, 2012.
13) http://netbeans.org/kb/docs/web/mysql-webapp.html
14) http://www.oracle.com/technetwork/java/index-jsp-135995.html

[Abstract]

A Java Application Development Platform with a Unified Modeling Language (UML) Plug-in (Part I)

Hiroshi NOTO

We have introduced a Java application development platform with a Unified Modeling Language tool plug-in in the computer labs of the Information Systems Center of Hokusei Gakuen University. The purpose of the set-up of our platform is that the students of our courses are able to manipulate visually their own logic in the domain modeling by trial and error on this platform to decide their business models and architecture specifications which are to result in their own applications. In our seminars and lectures, we have adopted Java as the programming language and NetBeans as an integrated development environment (IDE) in Java to take advantage of NetBeans' features to help develop Java applications efficiently. The basic design of our platform complex is presented. Each configuration of several combinations of platform components is elaborated on according to each course. We present case studies of actually running the platform in our courses.

Key words: Java Application, Database Management System, NetBeans IDE, Unified Modeling Language(UML), Reporting System