Global Journal of Computer Sciences: Theory and Research

Volume 8, Issue 3, (2018) 132-135

www.gjcs.eu

Developing java design patterns modeller with object-oriented programming

Egemen Tekkanat*, Kesan Yusuf Capraz Applied School, Trakya University, 22800, Edirne, Turkey
Murat Topaloglu, Kesan Yusuf Capraz Applied School, Trakya University, 22800, Edirne, Turkey

Suggested Citation:
Tekkanat, E. & Topaloglu, M. (2018). Developing java design patterns modeller with object-oriented programming. *Global Journal of Computer Sciences: Theory and Research*. *8*(3), 132–135.

Received from March 1, 2018; revised from July 23, 2018; accepted from November 22, 2018.
Selection and peer review under responsibility of Prof. Dr. Dogan Ibrahim, Near East University, Cyprus.

Abstract

Planning is a very important stage for developing software. A number of systems have been developed for the planning stage, which is a must for software development. Object-oriented programming plays the most important part technically in those systems. Using object-based programming effectively minimises the time and money spent in terms of software development processes. One of the best ways for this is to use design patterns that are also known as modules or program segments consisting of more than one classes and supporting code reuse in order to solve the problems recurring during the development of the software. The aim of this study is to develop a design patterns modeller to be used in Java programming language to make the software development a planned, secure and upgradable process while shortening the time spent and reducing the costs via good planning.

Keywords: Java, object-oriented, design patterns, modelling.

* ADDRESS FOR CORRESPONDENCE: **Egemen Tekkanat,** Kesan Yusuf Capraz Applied School, Trakya University, 22800, Edirne, Turkey. *E-mail address*: egementekkkanat@trakya.edu.tr / Tel.: +90-533-576-5806

1. Introduction

Design patterns are quite important in the field of software engineering and every design model deals with recurring problems (Ayata, 2010). Design patterns explain the relationships between the categories and objects in object-based programming (Ozbek, Ince, Turhan & Onder, 2014). Design patterns are methods that offer flexible, extendible and reusable solutions for the problems encountered during the development of software and they explain how to solve these problems in those situations (http://www.tasarimdesenleri.com/core/home.jsp, 2015). Moreover, design patterns increase maintainability, reusability and understandability of the system (Gamma, Helm, Johnson & Vlissides, 1994). Design patterns support the principle that is called Open-Close. This principle states that software should be open for extension and closed for modification (Meyer, 1988).

With increasing costs of projects in the field of informatics, the term 'quality' has become the main focus of most studies (Erdemir, Tekin & Buzluca, 2008). It is a recognised fact that software with high quality should also have high compatibility while being less complicated and dependent on others (Booch, 1991). Complexity is one of the basic problems that is associated with software development tools and methods (Khan & Khan, 2012).

There are lots of studies to examine the use of design patterns. Ali Bugdayci, in his master's thesis titled, 'Automated refactoring of design pattern implementations to aspect-oriented counterparts', concentrated the design patterns and image processing studies (Bugdayci, 2007). In addition, Kasim Sinan Yildirim, in his master's thesis titled 'Design of an object-oriented and real-time microkernel for embedded systems using design patterns' showed that design patterns can also be used in systematical works (Yildirim, 2006). The aim of this study is to gain more successful results during software development by making use of Java design templates.

OOP is a software programming model that uses objects which include the data and methods to create the behaviour expected from the software (Basaraner & Selcuk, 2007). IEEE defines OOP as a programming language model in which a system or software module is represented with objects and the links between these objects (IEEE Computer Society, 1983). The basic concepts of OOP are the abstraction, storage and class hierarchies that make developing, modifying and protecting big software easier and help us extend the programs with ease (Booch, 2006).

In today's computer world, many of the leading software companies have been supporting Java environment directly or indirectly with increasing amounts day by day (Inceoglu, 2004). Java platform has three different editions which are Java Micro Edition-JME (Gamma et al., 1994) for mobile phones, other hand and embedded devices, Java Standard Edition-JSE (Ozbek et al., 2014) and Java Enterprise Edition-JEE (Ayata, 2010; http://www.java.sun.com, 2015). In addition, Java programming language supports applet structures to run the written programs on the Internet (Inceoglu, 2004).

2. Materials and method

MVC and UML (Unified Modeling Language) templates were used for the development of the design template modeller application. UML is used for specification, visualisation and documentation of the units of the object-oriented systems that are being developed (Kurnaz, Cetin & Ince, 2003). UML class diagrams include four basic elements (Onel, Komesli & Okur, 2014), which are classes, relationships, object instances and packages (Zhang, Peng, Zhao & Li, 2008).

2.1. The development of the java design template modeler application

The application was developed in Java programming language. When the application runs, the files to be given will be in Java class format.

The aim of this study is to help the software developers integrate the available design templates into their future projects more comfortably. The software developer can select the template s/he wants to process. Then, the developer designs the empty template selected in the code panel in the application screen to build it into the application.

The application interface also includes a section for template information, which provides details for the templates. There is a panel in the application to show the templates that we added to our project.

There are two options in the Template Selection component of the application which are architectural templates and design templates.

The architectural templates have three options which are single-layered architecture, three-layer architecture and MVC architecture. When one of these options is selected, the related design code is added under the presentation, Business and Data layer in the code view section. These codes are created by the application in compliance with the selected architectural template. Template Information section gives a brief definition of the template that is selected.

Design Template includes 12 separate design templates, which are abstract factory, adapter, bridge, builder, composite, decorator, facade, factory method, flyweight, prototype and singleton. As in the architectural template, these templates have the presentation, Business and Data layers. When 'add selected template' button is clicked, the codes for the templates are created by the application. In addition to that, an information screen opens when 'view template information' is clicked.

3. Conclusion

Software of high quality means applications that work more productively, faster and fulfil your needs best. Thus, the higher the quality is, the higher the costs will be. Thanks to this application, the possible errors during runtime will be minimised using design templates for the design of the software, which will also save both time and money.

In order to make design templates more attractive, architectural roof-architectural template system, which is not covered within the design templates as a topic, can be included in the application.

The aim of this application is to extend the use of design templates while saving time, money and other resources. The application has a structure that is upgradeable, customisable and flexible at the same time. Finally, it is hoped that this application will contribute to the use of design templates more frequently in the field.

References

Ayata, M. (2010). *Effect of some software design patterns on real time software performance* (MSc Thesis). Ankara, Turkey: Middle East Technical University.

Basaraner, M. & Selcuk, M. (2007). Cok Ajanli Sistemler ve Kartografik Genellestirme. *Tmmob Harita ve Kadastro Muhendisleri Odasi, 11*, 2–6.

Bugdayci, A. (2007). *Automated refactoring of design pattern implementations to aspect oriented counterparts* (MSc Thesis). Ankara, Turkey: Middle East Technical University.

Booch, G. (1991). Object oriented design with applications. Redwood City, CA.

Booch, G. (2006). Object oriented analysis & design with application. India: Pearson Education India.

Erdemir, U., Tekin, U. & Buzluca, F. (2008). *Object oriented software metrics and software quality*. Yazilim Kalitesi ve Yazilim Gelistirme Araclari Sempozyumu.

Gamma, E., Helm, R., Johnson, R. & Vlissides, J. (1994). *Design patterns: elements of reusable object-oriented software*. Pearson Education.

http://www.tasarimdesenleri.com/core/home.jsp(2015). Retrieved May 20, 2015 http://www.tasarimdesenleri.com/core/home.jsp

http://www.java.sun.com. (2015). Retrieved May 20, 2015 http://www.java.sun.com

Inceoglu, M. M. (2004). *BOTE ogrencilerinin java programlama dili ogretimi konusundaki gorusleri.* Malatya Turkey: XIII. Ulusal Egitim Bilimleri Kurultayi.

IEEE Computer Society. (1983). Software engineering technical committee, *IEEE standard glossary of software engineering terminology, Institute of Electrical and Electronics Engineers.*

Khan, S. A. & Khan, R. A. (2012). Object oriented design complexity quantification model. *Procedia Technology, 4*, 548–554.

Kurnaz, S., Cetin, O. & Ince, F. (2003). Yazilim muhendisliginde kalite ve uml. *Havacilik ve Uzay Teknolojileri Dergisi, 1*(2), 1–12.

Meyer, B. (1988). *Object-oriented software construction* (vol 2, pp. 331–410), New York, NY: Prentice hall.

Ozbek, F., Ince, M., Turhan, M., & Onder, H. H. (2014). *E-universite icin esnek bir framework gelistirilmesi ve uygulanmasi*, Akademik Bilisim, Mersin Universitesi.

Onel, S., Komesli, M. & Okur, M. C. (2014). *UML ile modellenen cografi verilerin XSLT yardimiyla OWL'a donusturulmesi*, In: Proceedings of the 8th Turkish National Software Engineering Symposium (UYMS-2014), Guzelyurt, Turkey.

Yildirim, K. (2006). *Design of an object oriented and real-time microkernel for embedded systems using design patterns* (MSc Thesis). Izmir, Turkey: Ege University.

Zhang, C., Peng, Z. R., Zhao, T. & Li, W. (2008). Transformation of transportation data models from unified modeling language to web ontology language, *Transportation Research Record: Journal of the Transportation Research Board, 2064*(1), 81–89.