SHOULD BE REPLACED ON REQUIRED TITLE PAGE

Instruction

1. Open needed docx template (folder "title"/<your department or bach if bachelor student>.docx).

2. Put Thesis topic, supervisor's and your name in appropriate places on both English and Russian languages.

3. Put current year (last row).

4. Convert it to "title.pdf," replace the existing one in the root folder.

Contents

List of Tables

List of Figures

Abstract

Данная работа посвящена сравнительному анализу объектных моделей современных языков программирования, включая C#, C++, Golang, Java, Python, Ruby, JavaScript, Scala, Smalltalk и Zonnon. Актуальность работы обусловлена широким распространением объектно-ориентированной парадигмы и значительным разнообразием в ее реализации, что создает сложности для осознанного выбора технологий и архитектурных решений. Целью исследования является выявление сходств, различий, сильных и слабых сторон различных объектных моделей, а также разработка на этой основе предложений по созданию усовершенствованной модели. В результате исследования систематизированы теоретические основы объектных моделей, проведен детальный разбор их реализации в выбранных языках с выявлением областей применения и типичных ошибок, а также предложена авторская модель, интегрирующая лучшие практики. Практическая значимость работы заключается в том, что ее результаты могут быть использованы разработчиками для выбора языка и парадигм, архитекторами — для проектирования систем.

Chapter 1

Introduction

1.1 Spacing & Type

1.1.1 Creating a Subsection

Creating a Subsubsection

Creating a Subsubsection

Creating a Subsubsection

This is a heading level below subsubsection And this is a quote:

This is a table:

The package **upgreek** allows us to use non-italicized lower-case greek letters. See for yourself: $\upbeta$, $\boldsymbol{\upbeta}$, β, $\boldsymbol{\beta}$. Next is a numbered equation:

$$\|\boldsymbol{X}\|_{2,1} = \underbrace{\sum_{j=1}^{n} f_j(\boldsymbol{X})}_{\text{convex}} = \sum_{j=1}^{n} \|\boldsymbol{X}_{\cdot,j}\|_2 \tag{1.1}$$

Fig. 1.1. One kernel at x_s (*dotted kernel*) or two kernels at x_i and x_j (*left and right*) lead to the same summed estimate at x_s. This shows a figure consisting of different types of lines. Elements of the figure described in the caption should be set in italics, in parentheses, as shown in this sample caption.

The reference to equation (1.1) is clickable.

1.2 Theorems, Corollaries, Lemmas, Proofs, Remarks, Definitions,and Examples

Theorem 1. *Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.*

Proof. I'm a (very short) proof. □

Lemma 1. *I'm a lemma.*

Corollary 1. *I include a reference to Thm. 1.*

Proposition 1. *I'm a proposition.*

Remark. I'm a remark.

Definition 1. I'm a definition. I'm a definition. I'm a definition. I'm a definition. I'm a definition. I'm a definition. I'm a definition. I'm a definition. I'm a definition. I'm a definition. I'm a definition.

Example. I'm an example.

1.3 Section with linebreaks in the name

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

This is the second paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Chapter 2

Literature Review

2.1 The Concept and Components of the Object Model

The object model represents a fundamental framework that defines how a programming language represents and supports objects, classes, inheritance, encapsulation, polymorphism, and other related abstractions. According to commonly accepted terminology, the term object model has two interrelated meanings: (1) the properties of objects within a particular programming language, technology, notation, or methodology that employs these objects; and (2) a set of interfaces or classes through which a program can explore and manipulate specific aspects of its environment. Examples of object models include the Java Object Model, the Component Object Model (COM), and the Object Modeling Technique (OMT). Such object models are typically defined using concepts such as class, generic function, message, inheritance, polymorphism, and encapsulation.

Cardelli and Wegner in their work “On Understanding Types, Data Abstraction, and Polymorphism," defined object-oriented languages through three key requirements [1]. A language is considered object-oriented if it satisfies the following conditions:

- Support objects that represent data abstractions with an interface composed of named operations and an encapsulated internal state
- Objects in the language are associated with a specific object type
- Types may inherit attributes from their supertypes

These requirements were formulated in the form of a formula:
$object - oriented = dataabstractions + objecttypes + typeinheritance$
Booch, in his seminal work Object-Oriented Analysis and Design with Applications, describes the object model as a conceptual representation of the organized complexity of software systems [2].

> Booch (p. 40-41) concludes that the conceptual framework for anything object-oriented is the object model—a conceptual representation of the organized complexity of software. It consists of four major elements, i.e. abstraction, encapsulation, modularity, and hierarchy. In addition, the object model contains three minor element, i.e. typing, concurrency, and persistence.

This distinction between major and minor elements is fundamental: without the major elements, the model ceases to be object-oriented, whereas the minor elements are useful but not essential for supporting object orientation.

2.1.1 Major Components of the Object Model

Abstraction

Encapsulation

Modularity

Hierarchy

2.1.2 Minor Components of the Object Model

Typing

Concurrency

Persistence

2.1.3 The Selected Definition of the Object Model

Based on the conducted literature review, the following definition of the object model is adopted for the purposes of this work:

The object model is a conceptual framework that integrates a set of interrelated components — four major ones (abstraction, encapsulation, modularity, and hierarchy) and three minor ones (typing, concurrency, and persistence). It defines the means by which a programming language represents, organizes, and supports objects, classes, their interactions, inheritance relationships, mechanisms of polymorphism, and other related abstractions for the effective management of software system complexity.

Chapter 3

Methodology

Referencing other chapters 2, 3, 4, 5 and 6

. . .

Chapter 4

Implementation

. . .

Chapter 5

Evaluation and Discussion

...

Chapter 6

Conclusion

. . .

Bibliography cited

[1] L. Cardelli and P. Wegner, “On understanding types, data abstraction, and polymorphism,” *ACM Computing Surveys (CSUR)*, vol. 17, no. 4, pp. 471–523, 1985.

[2] G. Booch, R. A. Maksimchuk, M. W. Engle, B. J. Young, J. Connallen, and K. A. Houston, “Object-oriented analysis and design with applications,” *ACM SIGSOFT software engineering notes*, vol. 33, no. 5, pp. 29–29, 2008.

Appendix A

Extra Stuff

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Appendix B

Even More Extra Stuff

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.