

### Цели и задачи:

В рамках этой лабораторной работы необходимо показать общую архитектуру системы на двух уровнях: информационном и программном. Соответственно, сама работа также разбивается на две части.

### Концептуальная диаграмма классов:

Для концептуальной диаграммы классов необходимо составить словарь предметной области. Поскольку текстовое описание, которое было получено в рамках ЛР 1 и ЛР 4 (вспоминаем описание процесса), с большой вероятностью не является полным, предлагается выделить классы двумя способами последовательно.

Сначала необходимо выделить классы и их операции, используя метод Аббота. Для этого необходимо выделить в описании те фрагменты, которые будут отвечать за ту или иную характеристику модели (например, наличие класса или его составных частей, обобщение двух классов, их ассоциации и зависимости). Необязательно снова копировать и вставлять описание полностью, достаточно цитат с выделением нужной части и кратким пояснением, за что она отвечает. Обратите внимание, что при корректном выполнении прошлой лабораторной работы часть классов уже была выделена на диаграмме деятельности. Этот шаг даст те концептуальные классы, которые точно нужны в рамках системы и должны быть показаны на диаграмме классов.

После этапа анализа необходимо составить диаграмму классов, отражающую информационную архитектуру ИС. Что необходимо показать на этой диаграмме:

Концептуальные классы с атрибутами и операциями. При необходимости у атрибутов и операций покажите свойства, которые были рассмотрены на лекциях. Видимость атрибутов и операций здесь можно скрыть.

Ассоциации между классами, включая агрегации и композиции. Ассоциации должны показывать кратность на всех полюсах, возможность навигации и роль ассоциации на полюсах, свойства на полюсах, если они требуются, и подпись ассоциации с направлением чтения. При необходимости можно показать класс ассоциации.

Обобщения, если они необходимы. Если вы используете обобщения, покажите их подмножества.

Остальные элементы остаются на ваше усмотрение.

### Техническая диаграмма классов:

Техническая диаграмма классов создаётся строго после концептуальной, иначе теряется смысл проектирования. На технической диаграмме классов предлагается показать набор классов для возможной реализации ИС. Будем считать, что для реализации будет использован один из стандартных ООП-языков, например, Java или C#. В качестве архитектурного паттерна предлагается взять трёхслойную архитектуру, соответственно, необходимо показать набор классов для каждого из слоёв. При этом ограничения на архитектуру нет.

Для каждого слоя необходимо создать отдельную диаграмму классов с соответствующими пакетами.

После этого нужно сделать общую диаграмму пакетов, на которой будет показано, как они взаимодействуют друг с другом.

Что необходимо показать на этих диаграммах классов:

Классы и интерфейсы, по которым можно будет написать конкретный код. Здесь также нужны и атрибуты, и операции, но их расположение может меняться относительно концептуальной диаграммы классов.

Пакеты, которые группируют классы и интерфейсы в соответствии со слоями. Необязательно выделять ровно по одному пакету на слой, необязательно весь слой помещать в один пакет, можно, например, сделать несколько пакетов, отвечающих за инфраструктуру, и поместить их в один общий пакет.

Зависимости между классами с подходящими стереотипами (то есть не стоит везде использовать стереотип use).

Реализации интерфейсов.

Ассоциации между классами с агрегациями и композициями. Остальные украшения ассоциаций с концептуальной диаграммы классов можно не указывать.

Обобщения, если они необходимы, вместе с подмножествами.

Напомним, что диаграмма классов поддерживает и другие “надстройки”, которые могут помочь конкретно в вашей реализации.

## Решение:

### 1) Анализ прошлых работ и выделение классов

Из Use Case диаграмм и Activity диаграмм мы имеем:

Источник	Выявленные сущности
Use Case	Клиент, Кредитный менеджер, Старший менеджер, Заявка, Документы, Договор
Activity	CreditApplication, ScoringReport, VerificationResult, CreditAgreement, SignedCreditAgreement
Бизнес-процесс	Скоринговый отчет, Решение, Уведомление

### 2) Концептуальная диаграмма классов

Client
-clientId : string -firstName : string -lastName : string -passportNumber : string -phoneNumber : string -email : string -registrationDate : date
+submitApplication() +uploadDocuments() +signAgreement()

CreditApplication
-status : ApplicationStatus -riskLevel : RiskLevel -loanTerm : int -loanPurpose : string -loanAmount : decimal -applicationId : string -applicationDate : date
+calculateScoring() +changeStatus() +verifyDocuments()

Document
-documentId : string -type : DocumentType -fileName : string -uploadDate : date -verificationStatus : VerificationStatus
+verify() +validateFormat()

CreditManager
-employeeId : string -firstName : string -lastName : string -position : string -approvalLimit : decimal
+registerApplication() +analyzeApplication() +approveApplication() +requestAdditionDocuments()

SeniorManager
-employeeId : string -firstName : string -lastName : string -position : string -approvalLimit : decimal
+analyzeComplexApplication() +approveLargeApplication() +reviewCreditComitteeCases()

ScoringReport
-reportId : string -score : int -riskLevel : RiskLevel -generationDate : date -factors : string[]
+calculateRisk() +generateReport()

CreditAgreement
-agreementId : string -interestRate : decimal -signingDate : date -status : AgreementStatus -terms : string
+generateDocument() +sign() +validateTerms()

<<enumeration>> ApplicationStatus
NEW IN_PROGRESS APPROVED REJECTED CANCELLED

<<enumeration>> RiskLevel
LOW MEDIUM HIGH VERY_HIGH

<<enumeration>> DocumentType
PASSPORT INCOME_STATEMENT EMPLOYMENT_CONFIRMATION OTHER

<<enumeration>> VerificationStatus
PENDING VERIFIED REJECTED EXPIRED

