

|  |
| --- |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждениевысшего образования"МИРЭА - Российский технологический университет"РТУ МИРЭА |

**Институт** Информационных Технологий

**Кафедра** КИС

**ПРАКТИЧЕСКАЯ РАБОТА №5**

**Моделирование диаграмм в стиле ООП**

Студент группы: ИКБО-04-19 Сюртуков З.А *(Фамилия студента)*

Руководитель работы

*(Фамилия преподавателя)*

|  |  |  |
| --- | --- | --- |
|  |  |  |

Москва 2021

**Вступление**

**Цель работы:** научиться строить классовые диаграммы в стиле объектно-ориентированного программирования для различных систем

**Индивидуальное задание:** построить временную диаграмму для системы “Бюро находок”

**Диаграмма в стиле ООП**

Диаграммы в стиле ООП опираются на объектно-ориентированный стиль программирования, что означает подробное разбиение на различные классы, которые имеют внутри различные переменные, которые будут отличаться от объекта к объекту, и методы, позволяющие через их вызов объектам классов взаимодействовать между собой.

Объектно-ориентированное программирование (ООП) — методология программирования, основанная на представлении программы в виде совокупности объектов, каждый из которых является экземпляром определённого класса, а классы образуют иерархию наследования.

Идеологически ООП — подход к программированию как к моделированию информационных объектов, решающий на новом уровне основную задачу структурного программирования: структурирование информации с точки зрения управляемости, что существенно улучшает управляемость самим процессом моделирования, что, в свою очередь, особенно важно при реализации крупных проектов.

Основные принципы структурирования в случае ООП связаны с различными аспектами базового понимания предметной задачи, которое требуется для оптимального управления соответствующей моделью:

абстракция для выделения в моделируемом предмете важного для решения конкретной задачи по предмету, в конечном счёте — контекстное понимание предмета, формализуемое в виде класса;

инкапсуляция для быстрой и безопасной организации собственно иерархической управляемости: чтобы было достаточно простой команды «что делать», без одновременного уточнения как именно делать, так как это уже другой уровень управления;

наследование для быстрой и безопасной организации родственных понятий: чтобы было достаточно на каждом иерархическом шаге учитывать только изменения, не дублируя всё остальное, учтённое на предыдущих шагах;

полиморфизм для определения точки, в которой единое управление лучше распараллелить или наоборот — собрать воедино.

**Диаграмма в стиле ООП для системы “Бюро находок”**

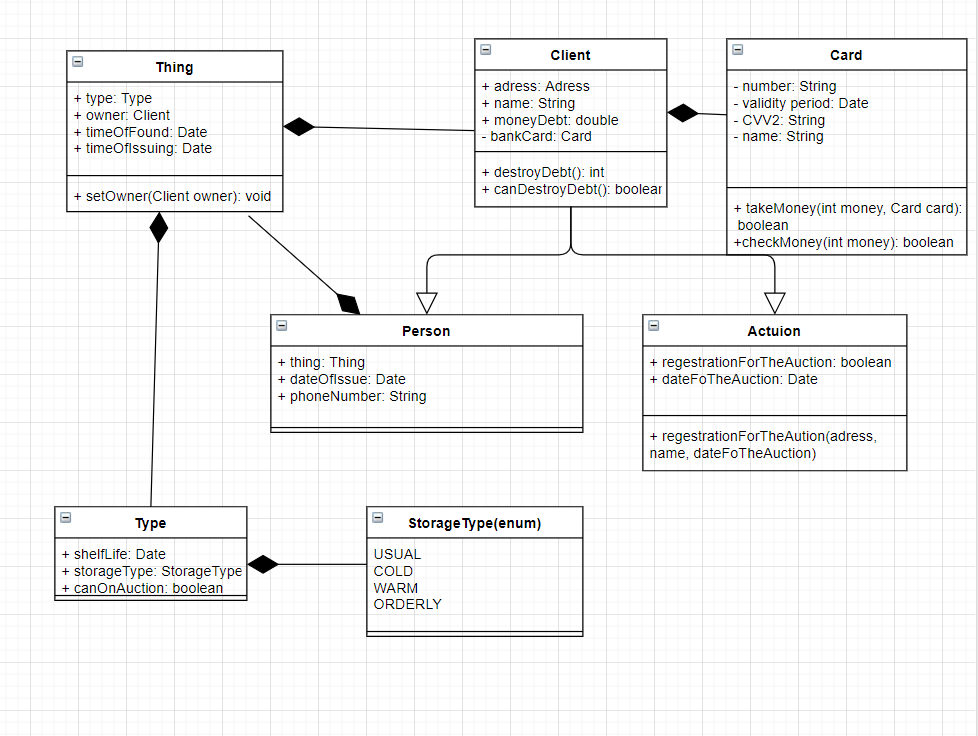
****

Рисунок 1. Диаграмма в стиле ООП для системы “Бюро находок”

**Класс Thing** – общий класс для всех хранимых в бюро вещей

**Переменные:**

Type type – определяет тип вещи

Client owner – определяет владельца вещи

Date timeOfFound – время нахождения вещи

Date timeOfIssuing – время выдачи вещи владельцу или аукциону

**Методы:**

void setOwner(Client owner) – установка владельца вещи, если она была кому-то или куда-то передана

**Класс Client** – общий класс для всех потенциальных владельцев вещи

**Переменные:**

Address address – адрес пребывания владельца

String name – наименование владельца

Double moneyDebt – долг за хранение утерянной вещи

Card bankcard – банковская карта для взимания долга

**Методы:**

int destroyDebt() – взимание долга с банковской карты, возвращает остаток долга

Boolean canDestroyDebt() – сравнивает баланс карты и имеющийся долг

**Класс Card** – класс банковской карты

**Переменные:**

String number – номер банковской карты

Date validityPeriod – срок действия

String CVV2 – секретный код

String name – имя владельца

**Методы:**

Boolean takeMoney(int money, Card card) – перевести сумму money на карты card. Возвращает информацию о том – успешно ли прошла операция

Boolean checkMoney(int money) – проверка наличия на карте суммы money

**Класс Person** – определяет человека, потерявшего вещь, наследуется от Client

**Переменные:**

Thing thing – потерянная вещь

Date dateOfIssue – дата выдачи вещи

String phoneNumber – телефонный номер

**Класс Auction** – определяет аукцион, на который выставляется утерянная вещь. Наследуется от класса Client

**Переменные:**

Boolean regestrationForTheAuction – была ли вещь зарегистрирована на аукционе

Date dateOfTheAuction – дата аукциона

**Методы:**

regestrationForTheAution(adress, name, dateForTheAuction) – регистрация аукциона.

**Класс Type** – тип вещи

**Переменные:**

Date shelfLife – срок хранения

StorageType storageType – тип хранения

Boolean canOnAuction – может ли вещь быть выставлена на аукцион