

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ В.Н. КАРАЗІНА
ФАКУЛЬТЕТ КОМП'ЮТЕРНИХ НАУК

КУРСОВА РОБОТА

з дисципліни «Об'єктно-орієнтоване програмування»

Тема «Моделювання роботи автосервісу»

Оцінка _____ балів / _____

Члени комісії:

_____ Богучарський С.І.

_____ Куклін В.М.

_____ Поклонський Є.В.

Виконав студент 2 курсу

групи КС-21

Синюк Валентин Миколайович

Керівник:

к.т.н., доцент Богучарський С.І.

РЕФЕРАТ

Відомості про об'єм курсової роботи: 24 ст., 11 рис., 9 джерел інформації.

Перелік ключових слів: МОДЕЛЮВАННЯ, АВТОСЕРВІС, ЦЕХ, ОБСЛУГОВУВАННЯ, ЩІЛЬНІСТЬ ПОТОКУ ЗАЯВОК, ЧЕРГА, МЕХАНІК, ЗАРОБІТНА ПЛАТА, ПРИБУТОК, СТАТИСТИКА, РЕКОМЕНДАЦІЇ.

Курсова робота, в своїй основі, передбачає розробку програмної моделі (ПМ) обслуговування автомобілів в цехах декотрого автосервісу. За допомогою ПМ описується діяльність автосервісу та здатність до: візуалізації, збирання статистики та надання рекомендацій. Автоматизація ПМ дозволяє досить якісно дослідити поведінку, з різними вхідними даними, та підрахувати необхідні величини.

В роботі розглядаються методи дослідження та інструментальні засоби, які використовувалися для візуалізації та реалізації цієї моделі. Окрім цього проведено формальне тестування продукту, за допомогою відлагоджувача програмного середовища.

ЗМІСТ

Список позначень та скорочень	4
Вступ.....	5
1 Постановка прикладної задачі	
1.1 Область визначення прикладного рішення	6
1.2 Постановка задачі предметної області.....	6
2 Методи та моделі реалізації прикладної задачі	
2.1 Інструментальні засоби та візуалізація	8
2.2 Модель програмного продукту	10
3 Засоби, інструменти та реалізація прикладного рішення	
3.1 Інструментальні засоби реалізації прикладної задачі	12
3.2 Побудова програмного продукту	13
3.3 Тестування програмного продукту	17
Висновок.....	23
Список джерел інформації.....	24

СПИСОК ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

КР – курсова робота;

ПМ – програмна модель;

ПП – програмний продукт;

ПЗ – програмне забезпечення;

ОС – операційна система;

ООП – об'єктно-орієнтоване програмування;

UML (англ. Unified Modeling Language) – уніфікована мова моделювання;

IDE (англ. Integrated Development Environment) – інтегроване середовище розробки;

JVM (англ. Java Virtual Machine) – віртуальна машина Java;

Javac (англ. Java compiler) – компілятор Java.

ВСТУП

В роботі описуватиметься процес розробки програмної моделі, по обслуговуванню автомобілів в цехах автосервісу. Головною метою моделювання є: надання рекомендацій, по управлінню автосервісом, на основі зібраних статистичних даних, враховуючи різні значення параметрів та випадкові величини.

Планується розробити модель, яка працюватиме в автономному режимі, за певним сценарієм, тобто конфігурації вхідних даних (параметрів) та методів (дій), які оброблятимуть їх. Для більшої складності та отримання достовірних результатів, заявки, на обслуговування автомобілів, будуть генеруватися випадковим чином та можуть містити декілька послуг.

Під час виконання програми, виконуватиметься візуалізація поточної ситуації в автосервісі, в реальному часі. Після закінчення, відбудеться збір та підрахунок даних, для надання статистики та рекомендацій. Статистика, буде містити: загальну кількість отриманих заявок, вдало та невдало наданих послуг різного виду; середній час обслуговування одного автомобіля; середню довжину черг в кожному цеху; середню зайнятість робітників і їхню заробітну плату, загальний прибуток та збитки автосервісу. Рекомендації, будуть містити поради, щодо менеджменту персоналу.

Структура курсової роботи складається зі вступу, списку позначень та скорочень, трьох розділів, висновку та списку джерел.

В першому розділі буде описана область визначення прикладного рішення та розкрита ціль, актуальність та новизна роботи. Також, буде розглянута поставлена задача предметної області.

У другому розділі буде розглянуто методи візуалізації та реалізації прикладної задачі. Для моделювання буде описано використання уніфікованої мови моделювання, а для реалізації прикладного рішення, будуть розглянуті різноманітні інструментальні засоби.

В третьому розділі буде описано, які саме інструментальні засоби, були використані в процесі розробки ПМ. Також, буде розглянута побудова та тестування програмного продукту. Даний розділ має практичну цінність.

1 ПОСТАНОВКА ПРИКЛАДНОЇ ЗАДАЧІ

1.1 Область визначення прикладного рішення

Область визначення прикладного рішення визначається: поставленою ціллю, актуальністю та новизною. Кожен з цих критеріїв, представлений нижче.

Ціль. Визначити оптимальне співвідношення кількості працівників в цехах автосервісу; виявити «вузькі» місця в його функціонуванні; надати рекомендації по керуванню автосервісом, на основі зібраних статистичних даних, а саме: як поліпшити алгоритм виконання роботи та які дії слід вжити, для уникнення ситуацій з втратою клієнтів, а, отже, і прибутку.

Актуальність. На теперішній час, існує тенденція до збільшення кількості фірм, які спеціалізуються на технічному обслуговуванні та ремонті автомобілів, тому, між ними, виникає конкуренція за володіння кількістю потенційних клієнтів. Пошук найкращих альтернатив, за деякими параметрами (витрати ресурсів і часу, ризики, наслідки прийнятого рішення), допоможуть керівникам цих фірм, володіти дійсно якісною інформацією для прийняття грамотних рішень, які безпосередньо мають вплив, на подальший розвиток і конкурентоспроможність фірми.

Новизна створюваної моделі полягає як у способі постановки задачі, так і в способі дослідження та ступені обґрунтування кінцевих результатів. Створення моделі ґрунтується на використанні вхідних даних, як даних, реального автосервісу. Тому в роботі будуть поєднуватися позитивні сторони використання аналітичного методу дослідження, що дає змогу отримати наближений розв'язок до поставленої задачі та експериментального, який дозволяє глибоко вивчити процеси розробки моделі і сконцентрувати увагу на тих параметрах, які представляють найбільший інтерес.

1.2 Постановка задачі предметної області

Автосервіс надає різні види послуг з ремонту та обслуговування автомобілів і включає в себе декілька цехів: техогляд, шиномонтаж, кузовний ремонт, ремонт двигуна. В кожному цеху працює декілька майстрів ($2 \leq K \leq 7$). Відомий тижневий розклад роботи автосервісу: 5 днів по 12 годин і два дні по 8 годин, без перерв.

Необхідно розробити імітаційну модель роботи автосервісу, при якій заявки на обслуговування автомобілів надходять випадковим чином, кожна заявка включає одну або декілька послуг. Кожна послуга виконується в певному цеху, відома середня тривалість її виконання і отримуваний при цьому прибуток.

Фактичний термін виконання заявки може відрізнятися від середнього на деяку випадкову величину, що змінюється в деякому діапазоні (наприклад, від години до декількох днів, в залежності від виду послуги). Випадковою величиною є також відрізок часу між послідовним появою двох заявок, вона має нормальний або рівномірний розподіл в деякому інтервалі (наприклад, від 15 хвилин до 1 години), причому щільність потоку заявок залежить від часу дня – в середині робочого дня заявки надходять частіше.

Надійшовши, заявки, утворюють черги, причому в загальному випадку заявка може зберігатися в черзі декілька днів. Один і той же автомобіль, може перебувати одночасно в декількох чергах, але його обслуговування в потрібних цехах виконується послідовно. Максимальний загальний термін обслуговування кожного автомобіля – тиждень, якщо ж після закінчення цього терміну ремонт (обслуговування) автомобіля ще не закінчений, то тоді власник забирає його з автосервісу, тим самим автосервіс втрачає своїх клієнтів, а, отже, і, прибуток.

Головною метою розробки даної моделі є визначення оптимального співвідношення числа робочих в цехах автосервісу; виявлення «вузьких» місць в його роботі (таких як нестача майстрів або їх простій), для збільшення прибутку. Тижнева зарплата кожного майстра визначається як 35% від принесеного ним прибутку автосервісу, але не менше 1 тис. грн. в день.

Період моделювання – тиждень, крок – М хвилин. Слід включити в параметри моделювання величини К і М, а також діапазони розкиду вищевказаних випадкових величин. Під час виконання програми, виконуватиметься візуалізація процесу, це передбачає показ поточної ситуації в автосервісі. Після закінчення, дана програма, виводитиме на екран статистичні дані. В них буде вказано: загальна кількість вдало обслужених автомобілів та наданих послуг різного виду; середній час обслуговування одного автомобіля; середня довжина черг в кожному цеху; середня зайнятість робітників і середня їхня зарплата, загальний прибуток.

2 МЕТОДИ ТА МОДЕЛІ РЕАЛІЗАЦІЇ ПРИКЛАДНОЇ ЗАДАЧІ

2.1 Інструментальні засоби та візуалізація

Проектування – застосовується на початку розробки програмного продукту. Найчастіше, розробники програмного забезпечення (ПЗ), схиляються до проектування візуальної моделі за принципами каскадної (водоспадної) моделі. Вона представляє собою послідовність розробки ПЗ. Ієрархія такої моделі складається з декількох основних стадій реалізації та впровадження: складання вимог до програмного продукту (ПП), проектування та реалізація програмної моделі (ПМ), верифікація, тестування та технічне обслуговування ПП. Розглянемо етап проектування ПМ.

Інструментальні засоби, які використовуються для проектування, тобто моделювання та візуалізації ПМ, ґрунтуються на UML – уніфікованій мові моделювання. На теперішній день, вже існує досить багато редакторів та засобів для створення UML діаграм. Найпоширенішими програмами є: Microsoft Visio, Visual Paradigm, Eclipse IDE, IntelliJ IDEA. Наведемо коротку характеристику цих програмних середовищ, в яких реалізуються UML діаграми:

1 Microsoft Visio – векторний графічний редактор, редактор діаграм і блок-схем для Windows.

2 Visual Paradigm – це інструмент, для створення UML діаграм, що підтримує UML та предметно-орієнтовану мову моделювання систем SysML. Крім підтримки моделювання, він забезпечує можливості генерації звітів та коду.

3 Eclipse IDE – середовище розробки ПЗ, в якому можна додатково встановити модуль (плагін) для створення UML діаграм, наприклад, «The ObjectAid UML Explorer for Eclipse». Цей плагін надає особливі компоненти, які автоматично з'єднують моделі UML з вихідним кодом.

4 IntelliJ IDEA – середовище розробки ПЗ, з можливістю автоматичної генерації UML діаграм. Але ця особливість, у відмінності від інших інструментів, наявна тільки у версії Ultimate.

Для проектування візуальної моделі, яка зображена на рисунку 2.1, використовувався інструмент з середовища розробки програмного забезпечення

Eclipse, а саме додатково встановлений модуль (плагін) – «The ObjectAid UML Explorer for Eclipse».

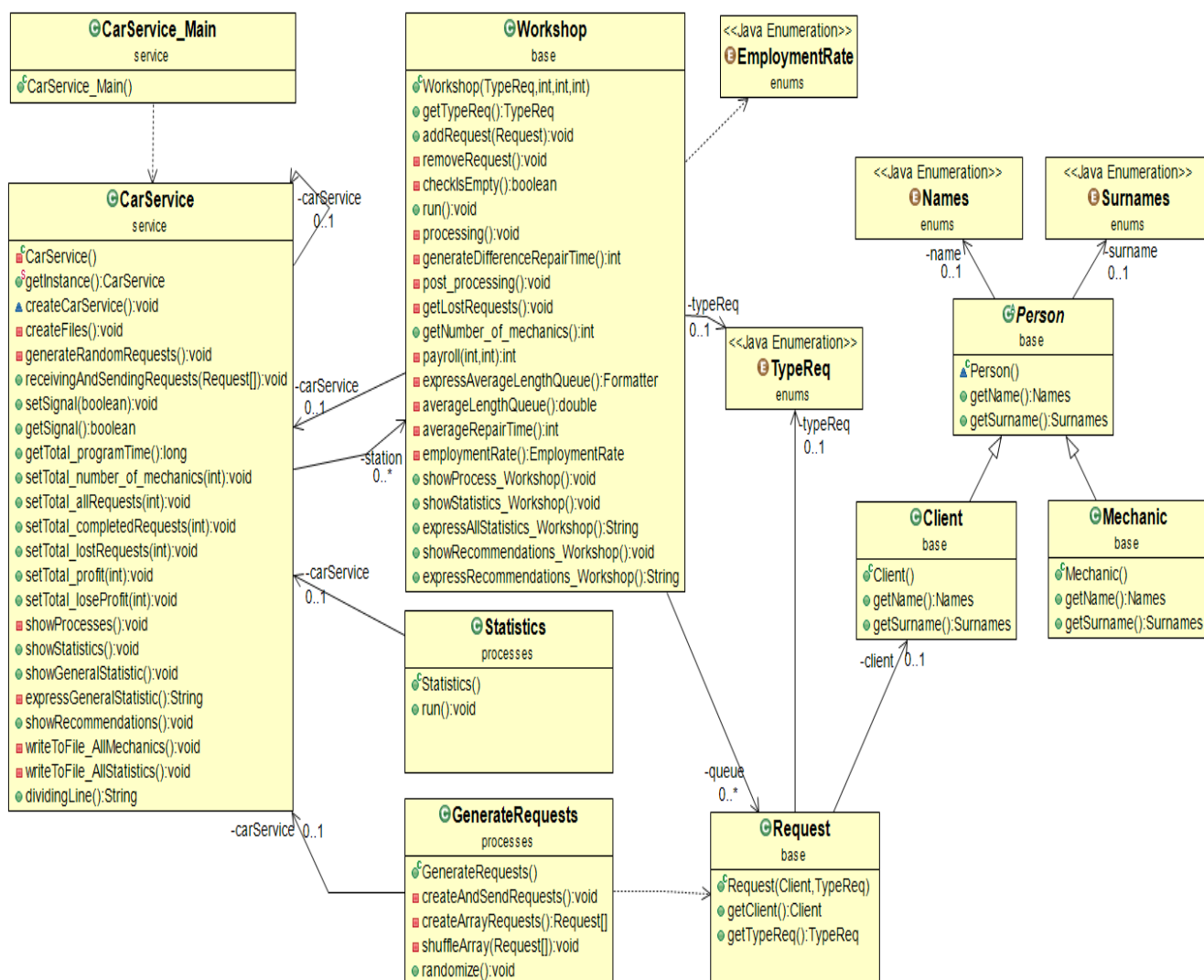


Рисунок 2.1 – UML діаграма рішення ПМ

Це легкий та гнучкий інструмент для графічного представлення існуючого коду. За допомогою нього, можна створювати діаграми класів та послідовностей, які автоматично оновлюються не виходячи з синхронізації. Автоматична генерація діаграм з вихідного коду та вбудований функціонал, дозволяють якісно та швидко відтворювати модель прикладної задачі.

Плагін дуже простий у використанні. Управління відбувається за допомогою контекстного меню, при натисканні правої кнопки миші.

2.2 Модель програмного продукту

Модель ПП буде розроблятися на такій мові програмування, як Java. Дана мова програмування дає змогу реалізувати основні парадигми об'єктно-орієнтованого програмування (ООП): абстракція, інкапсуляція, успадкування та поліморфізм.

Розглянемо ієрархію пакетів ПП, яку зображено на рисунку 2.2:

- 1) пакет «base» – включає в себе класи, на яких базуються інші класи ПП;
- 2) пакет «enums» – містить перерахування логічно пов'язаних констант;
- 3) пакет «files» – містить декілька файлів, для запису вихідних даних;
- 4) пакет «processes» – охоплює класи, в яких відбувається процес генерації заявок та робота зі статистикою;
- 5) пакет «service» – головний пакет, оскільки містить клас, в якому наявна точка входу в програму та клас, який представляє собою сам автосервіс.

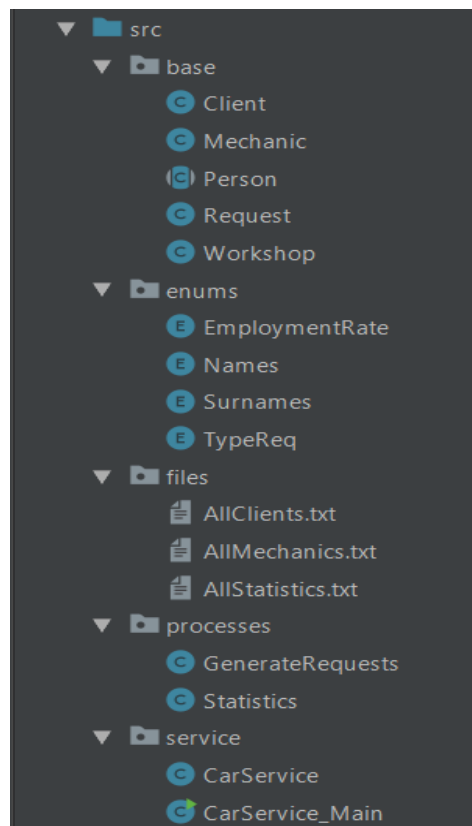


Рисунок 2.2 – Ієрархія пакетів рішення ПП

Коротко розглянемо структуру ПП, зображену на діаграмі класів:

1 Клас `CarService_Main` – основний клас, оскільки містить в собі точку входу в програму. Також, в ньому створюється об'єкт типу класу `CarService`.

2 Клас `CarService` – прототип автосервісу. Для цього реалізовується шаблон проектування – `Singleton`. Клас відповідає за всю функціональність автосервісу та зв'язки між компонентами ПМ, тобто, займається обробкою даних таких класів, як: `Workshop`, `GenerateRequests`, `Statistics`.

3 Клас `GenerateRequests` – відповідає за утворення заявок з випадковим переліком послуг та проміжком звернень клієнтів до автосервісу. Даний клас містить поле класу `Request`, тому, він має здатність на керування даними цього класу.

4 Клас `Workshop` – клас, який є зразком для проектування та функціонування кожного окремого цеху. Тип цеху визначається за допомогою перерахування `TypeReq`. Для роботи в заявками, клас містить моногенну колекцію, у виді черги з об'єктів класу `Request`.

5 Клас `Statistics` – клас, який необхідний для збору даних про виконання роботи кожного окремого цеху автосервісу та надання обґрунтованих рекомендацій.

6 Клас `Request` – клас, який представляє собою зразок заявки, яка надходить від клієнта до автосервісу. За інформацію про клієнта дбає клас `Client`, а за тип заявки – перерахування `TypeReq`.

7 Клас `Person` – абстрактний клас. Похідні від нього класи: `Client` та `Mechanic`. Для ініціалізації своїх полів, використовуються перерахування `Names` і `Surnames`.

3 ЗАСОБИ, ІНСТРУМЕНТИ ТА РЕАЛІЗАЦІЯ ПРИКЛАДНОГО РІШЕННЯ

3.1 Інструментальні засоби реалізації прикладної задачі

Виконання програм на Java базується на використанні важливого елементу мови – віртуальної машини (JVM), яка є основою виконуючої системи Java, так званої Java Runtime Environment (JRE). JVM виконує байт-код, попередньо створений з вихідного тексту Java-програми компілятором Java (Javac). Компілятор приймає вихідний код, відповідний специфікації Java Language Specification і повертає байт-код, вже відповідний специфікації Java Virtual Machine Specification.

Одна з переваг такого підходу є кросплатформність – властивість ПЗ працювати більш ніж на одній програмній (в тому числі – операційній системі (ОС)) або апаратній платформі. Кросплатформність дозволяє суттєво скоротити витрати на розробку нового або адаптацію існуючого ПЗ.

До важливих та корисних компонентів програмування на Java, також, відноситься віртуалізація – надання набору обчислювальних ресурсів або їх логічне об'єднання, абстраговане від механічної реалізації, що виконуються на одному фізичному ресурсі. Для реалізації цього механізму, використовується гіпервізор – комп'ютерна програма або обладнання процесора, що забезпечує одночасне і паралельне виконання декількох JVM, на кожній з яких виконується власна ОС, на одному фізичному комп'ютері. В якості прикладів можна виділити такі гіпервізори, як: «vSphere Hypervisor», «VMware ESXi».

Другорядним, але не менш важливим елементом програмування на Java є – середовище розробки ПЗ (IDE), яке необхідне для реалізації прикладних задач та потребує ретельного підбору, вивчення та ознайомлення. Зважаючи на всі переваги та недоліки багатьох широкодоступних IDE, був визначений гідний кандидат, як інструмент, що використовуватиметься – IntelliJ IDEA версії Ultimate. Він представляє собою інтегроване середовище розробки ПЗ для багатьох мов програмування, розроблене компанією JetBrains. Існують дві версії цієї IDE:

- а) Community Edition – безкоштовна версія з урізаним функціоналом;
- б) Ultimate Edition – комерційна повнофункціональна версія.

Порівняємо та надамо стислу характеристику цих версій. Більшість користувачів даної IDE використовують саме безкоштовну версію. Вона підтримує велику кількість бажаних інструментів для розробки ПЗ, наприклад, такі мови програмування як: Java, Groovy, Kotlin, Scala; такі засоби складання як: Gradle, Maven; та такі системи контролю версій як: Git, SVN, Mercurial.

Комерційна версія відрізняється наявністю підтримки багатьох інших важливих структур та технологій JVM, включаючи: Hibernate, Guice, FreeMarker, Velocity, Thymeleaf, Struts тощо. Також, придбавши цю версію, з'являться інструменти для роботи з базами даних, SQL.

3.2 Побудова програмного продукту

В результаті розробки ПП, виконано все, що було заплановано та затребувалося прикладною задачею, критерії по реалізації ПП дотримані, характеристика усім компонентам надана, всі можливі результати – враховані. Тому отримана структура ПП є коректною та складається з наступних класів, з переліками основних методів:

1 Клас `CarService_Main` – основний клас, оскільки містить в собі точку входу в програму. Також, в ньому створюється об'єкт типу класу `CarService`.

2 Клас `CarService` – прототип автосервісу. В класі реалізовується потокобезпечний шаблон проектування – `Singleton`. Тому об'єкт класу доступний у всій програмі та в одному екземплярі. В ньому реалізовані основні функції для роботи з автосервісом. Також, він володіє візуалізаційною здатністю та інфографікою. Основними методами, окрім аксесорів, є:

а) `getInstance()` – метод, для отримання єдиного об'єкта класу;

б) `createCarService(CarService carService)` – метод, який зображено на рисунку 3.1, необхідний для створення автосервісу з масиву цехів та запуску потоків на виконання;

в) `createFiles()` – метод, для створення або перезапису файлів;

г) `generateRandomRequests()` – метод, для генерації раптових заявок;

д) `receivingAndSendingRequests(Request[] requests)` – метод, для отримання готового масиву необхідних послуг однієї заявки, та відправка її на

обробку, в потрібний цех;

е) showProcesses() – метод, для візуалізації на консолі, поточного стану автосервісу, в реальному часі;

ж) showStatistics() – метод, для візуалізації на консолі, зібраних статистичних даних автосервісу;

и) showGeneralStatistic() – метод, для візуалізації на консолі, загальної статистики автосервісу;

і) showRecommendations() – метод для візуалізації на консолі рекомендацій, по управлінню автосервісом, на основі зібраних статистичних даних;

к) writeToFile_AllMechanics() – метод, для запису в файл, всіх механіків кожного окремого цеху автосервісу;

л) writeToFile_AllStatistics() – метод, для запису в файл, всієї статистики та рекомендацій, для кожного окремого цеху.

```
void createCarService() throws IOException, InterruptedException {
    station[0] = new Workshop(TypeReq.Техосмотр, repair_cost: 1500, maxRepairTime: 420, difference_V: 360);
    station[1] = new Workshop(TypeReq.Шиномонтаж, repair_cost: 3000, maxRepairTime: 560, difference_V: 720);
    station[2] = new Workshop(TypeReq.Кузовной_ремонт, repair_cost: 4500, maxRepairTime: 770, difference_V: 1080);
    station[3] = new Workshop(TypeReq.Ремонт_двигателя, repair_cost: 7500, maxRepairTime: 910, difference_V: 1440);

    createFiles();
    writeToFile_AllMechanics(); // write to file "AllMechanics.txt"
    generateRandomRequests(); // generate requests
    new Statistics(); // run threat class Statistics
    writeToFile_AllStatistics(); // write to file "AllStatistics.txt"
}
```

Рисунок 3.1 – Створення автосервісу

3 Клас GenerateRequests – відповідає за утворення заявок з випадковим переліком послуг. Даний клас містить такі методи як:

а) createAndSendRequests() – метод, для створення і відправки масиву заявок, в подальшу обробку;

б) `Request[] createArrayRequests()` – метод, в якому відбувається конструювання набору послуг, кожного клієнта, що звернувся до автосервісу. Метод повертає заповнений масив заявок, з даними про клієнта, за допомогою обробки даних з класів `Request` та `Client`;

в) `randomize()` – метод, який зображено на рисунку 3.2, необхідний для генерування запитів на формування масиву заявок, за графіком робочого тижня.

```
public void randomize() throws InterruptedException, IOException {
    for (int day = 1; day <= 7; day++) {
        int step;

        if (day <= 5) { /* для будних днів */
            for (int minute = 0; minute <= WEEKDAY; minute += step) {
                if (minute <= 270 | minute >= 450) {
                    step = random.nextInt( bound: 31) + 30;
                } else { /* середина дня (заявки поступають частіше) */
                    step = random.nextInt( bound: 16) + 15;
                }
                createAndSendRequests();
                Thread.sleep( millis: step * 10);
            }
        } else { /* для вихідних днів */
            for (int minute = 0; minute <= OFFDAY; minute += step) {
                if (minute <= 180 | minute >= 300) {
                    step = random.nextInt( bound: 31) + 30;
                } else {
                    step = random.nextInt( bound: 16) + 15;
                }
                createAndSendRequests();
                Thread.sleep( millis: step * 10);
            }
        }
    }
    carService.setSignal(false); // завершення генерування запитів
}
```

Рисунок 3.2 – Генерування запитів на формування масиву заявок, за графіком робочого тижня

4 Клас `Workshop` – клас, який є зразком, для проектування та функціонування кожного окремого цеху. Отримання даних для роботи, відбувається визначенням в класі конструктором. Цей клас розширює вбудований клас `Thread`, який в свою чергу, інкапсулює стандартні механізми роботи з потоками. Також, важливим та

обов'язковим критерієм, було створення моногенної колекції у виді черги з об'єктів класу Request. Окрім аксесорів, в класі представлені такі головні методи як:

- а) `addRequest(Request request)` – метод, для додавання заявки на обробку;
- б) `removeRequest()` – метод, для видалення заявки з черги, після завершення обробки;
- в) `run()` – метод, для запуску потоку обробки даних;
- г) `processing()` – метод, який зображено на рисунку 3.3, необхідний для обробки заявки;
- д) `post_processing()` – метод, для пост-обробки даних цеху і відправка їх аксесорам цього класу та класу CarService;
- е) `payroll()` – метод, для підрахунку заробітної плати механіків;
- ж) `showProcess_Workshop()` – метод, для візуалізації на консолі, поточного стану цеху;
- и) `showStatistics_Workshop()` – метод, для візуалізації на консолі зібраних статистичних даних цеху;
- к) `showRecommendations_Workshop()` – метод, для візуалізації на консолі рекомендацій з управління цехом.

```
private void processing() throws InterruptedException {
    int tempV = (repair_time + generateDifferenceRepairTime()) * 10; // время обслуживания
    sleep(tempV);

    worktime += tempV; // подсчитываем время работы
    total += repair_cost; // увеличиваем прибыль
    completedRequests++; // указываем что заявка обработана
    removeRequest(); // удаляем выполненную заявку
}
```

Рисунок 3.3 – Обробка заявки в цеху

5 Клас Statistics – клас, необхідний для збору даних про виконання роботи кожного окремого цеху автосервісу та надання обґрунтованих рекомендацій.

Отримання даних для роботи, відбувається визначенням в класі конструктором. Також, клас розширює вбудований клас Thread, який в свою чергу, інкапсулює стандартні механізми роботи з потоками. Цей клас містить тільки один метод – run(), який потрібен для запуску потоку обробки даних.

6 Клас Request – клас, який представляє собою зразок заявки, яка надходить від клієнта до автосервісу. В класі присутні тільки два аксесори:

а) Client getClient() – метод, для отримання об'єкта типу класу Client;

б) TypeReq getTypeReq() – метод, для отримання константи типу перерахування TypeReq.

7) клас Person – абстрактний клас. Похідні від нього класи: Client та Mechanic. Клас має тільки два аксесори:

а) Names getName() – метод, для отримання константи типу перерахування Names;

б) Surnames getSurname() – метод, для отримання константи типу перерахування Surnames.

3.3 Тестування програмного продукту

Процес функціонування ПП, абстрактно, можна поділити на декілька стадій:

1 Створення автосервісу та ініціалізація відповідних полів.

2 Виконання роботи по генерації заявок та їх паралельне оброблення в відповідних цехах з візуалізацією процесів на консолі IDE.

3 Отримання кінцевих даних у формі статистики та рекомендацій. Вихідні дані записуються у співвідносний файл для статистики.

Протестуємо ПП на виявлення неточностей, помилок та багів, в кожній з виділених стадій окремо, за допомогою відлагоджувача програмного середовища IntelliJ IDEA.

Перша стадія представляє собою задання вхідних даних, які передаються до конструктору, кожного окремого цеху. За це відповідає метод для створення автосервісу з класу CarService, який був зображений вище, на рисунку 3.1. Задаються такі дані як: тип цеху, встановлена вартість обслуговування,

встановлений час виконання послуги та різниця від встановленого часу. Ці дані легко змінюються під вузькоспрямовані потреби. Візуалізація непередбачена.

Розроблена модель працює в автономному режимі, тому на другій стадії функціонування ПП, розпочнуть свою роботу основні процеси. До них належать: генерація раптових заявок та оброблення їх в відповідних цехах, враховуючи різні значення параметрів та випадкові величини, а також, передбачена візуалізація поточної ситуації в автосервісі, як зображено на рисунку 3.4.

```

Run: CarService_Main x
"C:\Program Files\Java\jdk1.8.0_192\bin\java.exe" ...

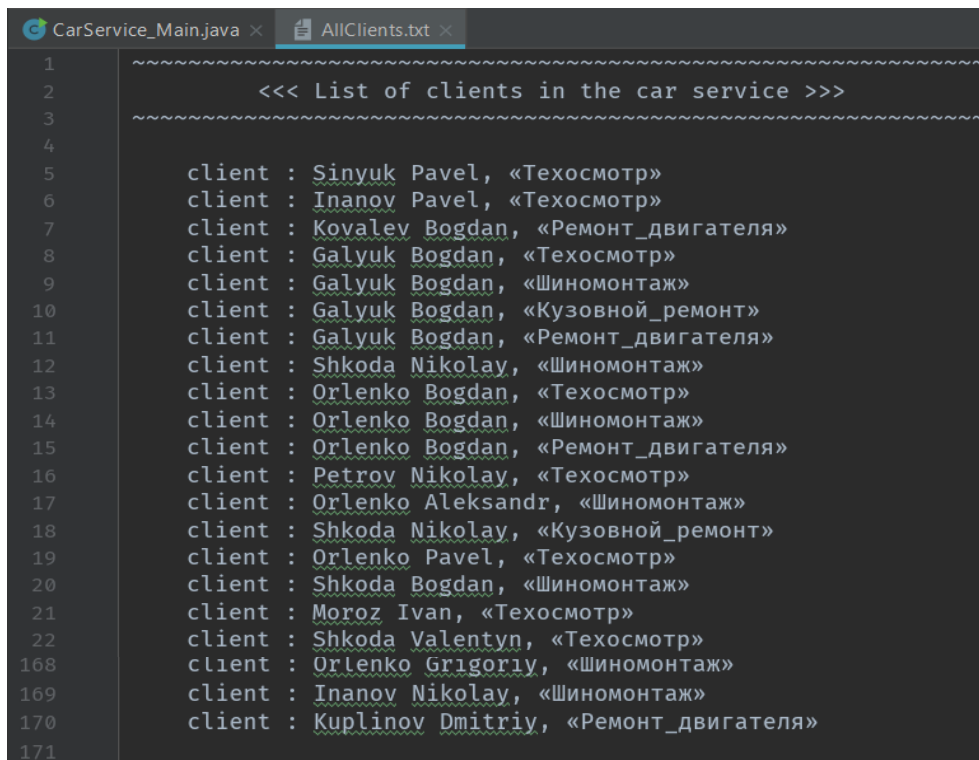
тип цеха: allR => compR ∞ averageT => rate
● Цех - «Техосмотр»: 1 => 0 ∞ 42 min. => LOW
● Цех - «Шиномонтаж»: 1 => 0 ∞ 56 min. => LOW
● Цех - «Кузовной_ремонт»: 0 => 0 ∞ 77 min. => LOW
● Цех - «Ремонт_двигателя»: 0 => 0 ∞ 91 min. => LOW
=====
тип цеха: allR => compR ∞ averageT => rate
● Цех - «Техосмотр»: 1 => 0 ∞ 42 min. => LOW
● Цех - «Шиномонтаж»: 1 => 0 ∞ 56 min. => LOW
● Цех - «Кузовной_ремонт»: 0 => 0 ∞ 77 min. => LOW
● Цех - «Ремонт_двигателя»: 1 => 0 ∞ 91 min. => LOW
=====
тип цеха: allR => compR ∞ averageT => rate
● Цех - «Техосмотр»: 1 => 1 ∞ 210 min. => LOW
● Цех - «Шиномонтаж»: 1 => 1 ∞ 140 min. => LOW
● Цех - «Кузовной_ремонт»: 0 => 0 ∞ 77 min. => LOW
● Цех - «Ремонт_двигателя»: 2 => 1 ∞ 455 min. => LOW
=====
тип цеха: allR => compR ∞ averageT => rate
● Цех - «Техосмотр»: 1 => 1 ∞ 210 min. => LOW
● Цех - «Шиномонтаж»: 1 => 1 ∞ 140 min. => LOW
● Цех - «Кузовной_ремонт»: 0 => 0 ∞ 77 min. => LOW
● Цех - «Ремонт_двигателя»: 3 => 2 ∞ 455 min. => LOW
=====
тип цеха: allR => compR ∞ averageT => rate
● Цех - «Техосмотр»: 2 => 1 ∞ 210 min. => LOW
● Цех - «Шиномонтаж»: 1 => 1 ∞ 140 min. => LOW
● Цех - «Кузовной_ремонт»: 0 => 0 ∞ 77 min. => LOW
● Цех - «Ремонт_двигателя»: 3 => 2 ∞ 455 min. => LOW
=====
тип цеха: allR => compR ∞ averageT => rate
● Цех - «Техосмотр»: 3 => 2 ∞ 210 min. => LOW

```

Рисунок 3.4 – Візуалізація поточної ситуації в автосервісі

Окрім цього, відбуваються процеси запису даних, до відповідних файлів, для

клієнтів та працюючих механіків автосервісу, це зображено на рисунках 3.5 та 3.6 відповідно.

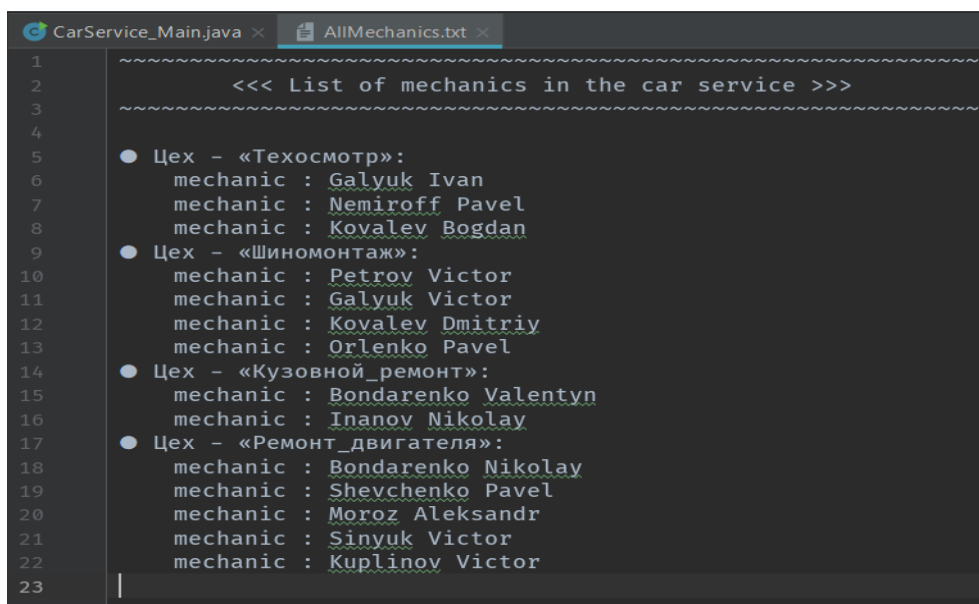


```

1  ~~~~~
2  <<< List of clients in the car service >>>
3  ~~~~~
4
5  client : Sinyuk Pavel, «Техосмотр»
6  client : Inanov Pavel, «Техосмотр»
7  client : Kovalev Bogdan, «Ремонт_двигателя»
8  client : Galyuk Bogdan, «Техосмотр»
9  client : Galyuk Bogdan, «Шиномонтаж»
10 client : Galyuk Bogdan, «Кузовной_ремонт»
11 client : Galyuk Bogdan, «Ремонт_двигателя»
12 client : Shkoda Nikolay, «Шиномонтаж»
13 client : Orlenko Bogdan, «Техосмотр»
14 client : Orlenko Bogdan, «Шиномонтаж»
15 client : Orlenko Bogdan, «Ремонт_двигателя»
16 client : Petrov Nikolay, «Техосмотр»
17 client : Orlenko Aleksandr, «Шиномонтаж»
18 client : Shkoda Nikolay, «Кузовной_ремонт»
19 client : Orlenko Pavel, «Техосмотр»
20 client : Shkoda Bogdan, «Шиномонтаж»
21 client : Moroz Ivan, «Техосмотр»
22 client : Shkoda Valentyn, «Техосмотр»
168 client : Orlenko Grigoriy, «Шиномонтаж»
169 client : Inanov Nikolay, «Шиномонтаж»
170 client : Kuplinov Dmitriy, «Ремонт_двигателя»
171

```

Рисунок 3.5 – Записані дані до файлу для інформації про клієнтів



```

1  ~~~~~
2  <<< List of mechanics in the car service >>>
3  ~~~~~
4
5  ● Цех - «Техосмотр»:
6    mechanic : Galyuk Ivan
7    mechanic : Nemiroff Pavel
8    mechanic : Kovalev Bogdan
9  ● Цех - «Шиномонтаж»:
10   mechanic : Petrov Victor
11   mechanic : Galyuk Victor
12   mechanic : Kovalev Dmitriy
13   mechanic : Orlenko Pavel
14  ● Цех - «Кузовной_ремонт»:
15   mechanic : Bondarenko Valentyn
16   mechanic : Inanov Nikolay
17  ● Цех - «Ремонт_двигателя»:
18   mechanic : Bondarenko Nikolay
19   mechanic : Shevchenko Pavel
20   mechanic : Moroz Aleksandr
21   mechanic : Sinyuk Victor
22   mechanic : Kuplinov Victor
23

```

Рисунок 3.6 – Записані дані до файлу для інформації про механіків

Остання, завершуюча стадія функціонування ПП, містить процеси збору та

підрахунку даних, необхідних для надання відповідної загальної статистики автосервісу та кожного його окремого цеху, окрім цього, надаються рекомендації по управлінню цехами автосервісу. Для наглядного подання інформації, була передбачена візуалізація даних на консолі та виведення цих даних у співвідносні файли.

Статистика, окремо взятого цеху, яка зображена на рисунку 3.7, містить: загальну кількість отриманих заявок, кількість вдало та невдало наданих послуг; середній час обслуговування одного автомобіля; середню довжину черги; зайнятість робітників і їхню заробітну плату, загальний і чистий прибутки та можливі збитки цеху. Рекомендації, які зображено на рисунку 3.9, містять поради, щодо менеджменту персоналу кожного окремого цеху автосервісу.

```

CarService_Main.java x AllStatistics.txt x
1 ~~~~~
2 <<< All statistics the car service >>>
3 ~~~~~
4
5 <<< Цех - «Техосмотр»
6 <<< Механиков - 3
7   ● Общее число заявок: 45
8   ● Обслуженно заявок : 28
9   ● Не будет обслуженно ≈ 0
10  ● Средняя длина очереди: 1.61
11  ● Фиксированное время обслуживания: 140 min.
12  ● Среднее время обслуживания: 160 min.
13  ● Время полезной нагрузки: 4480 min.
14  ● Время простоя цеха: 80 min.
15  ● Занятость рабочих: HIGH
16  ● Доход цеха: 42000€
17  ● Зарплата механика: 7000€
18  ● Чистая прибыль цеха: 21000€
19  ● Потерянный доход ≈ 0€
20 ~~~~~
21 <<< Цех - «Шиномонтаж»
22 <<< Механиков - 4
23   ● Общее число заявок: 53
24   ● Обслуженно заявок : 25
25   ● Не будет обслуженно ≈ 0
26   ● Средняя длина очереди: 2.12
27   ● Фиксированное время обслуживания: 140 min.
28   ● Среднее время обслуживания: 149 min.
29   ● Время полезной нагрузки: 3744 min.
30   ● Время простоя цеха: 816 min.
31   ● Занятость рабочих: HIGH
32   ● Доход цеха: 75000€
33   ● Зарплата механика: 7000€
34   ● Чистая прибыль цеха: 47000€
35   ● Потерянный доход ≈ 0€
36 ~~~~~
37 <<< Цех - «Кузовной_ремонт»
38 <<< Механиков - 2
39   ● Общее число заявок: 32
40   ● Обслуженно заявок : 17
41   ● Не будет обслуженно ≈ 4
42   ● Средняя длина очереди: 1.88
43   ● Фиксированное время обслуживания: 385 min.
44   ● Среднее время обслуживания: 268 min.
45   ● Время полезной нагрузки: 4560 min.
46   ● Время простоя цеха: 0 min.
47   ● Занятость рабочих: HIGH
48   ● Доход цеха: 76500€
49   ● Зарплата механика: 13387€
50   ● Чистая прибыль цеха: 49726€
51   ● Потерянный доход ≈ 18000€
52 ~~~~~
53 <<< Цех - «Ремонт_двигателя»
54 <<< Механиков - 5
55   ● Общее число заявок: 36
56   ● Обслуженно заявок : 19
57   ● Не будет обслуженно ≈ 0
58   ● Средняя длина очереди: 1.89
59   ● Фиксированное время обслуживания: 182 min.
60   ● Среднее время обслуживания: 223 min.
61   ● Время полезной нагрузки: 4251 min.
62   ● Время простоя цеха: 309 min.
63   ● Занятость рабочих: HIGH
64   ● Доход цеха: 142500€
65   ● Зарплата механика: 9975€
66   ● Чистая прибыль цеха: 92625€
67   ● Потерянный доход ≈ 0€

```

Рисунок 3.7 – Записані дані статистики для кожного окремого цеху до файлу
статистики

Загальна статистика автосервісу, яка зображена на рисунку 3.8, містить: кількість механіків автосервісу, загальну кількість отриманих заявок, кількість вдало та невдало наданих послуг, загальний прибуток та можливі збитки автосервісу.

```

69 ~~~~~
70 <<< General statistic in the car service >>>
71 ~~~~~
72
73 ● Механиков в автосервисе: 12
74 ● Всего заявок получено: 166
75 ● Обработано заявок: 61
76 ● Не обработано заявок ≈ 4
77 ● Доход автосервиса: 210351€
78 ● Потерянный доход ≈ 18000€
79
80 ~~~~~

```

Рисунок 3.8 – Записані дані загальної статистики до файлу статистики

Рекомендації, які зображено на рисунку 3.9, містять поради, щодо менеджменту персоналу кожного окремого цеху автосервісу.

```

81 ~~~~~
82 <<< Recommendations for the car service >>>
83 ~~~~~
84
85 <<< Цех «Техосмотр» – не несёт убытки.
86 ● Цех работает эффективно! Уровень занятости: HIGH
87
88 <<< Цех «Шиномонтаж» – не несёт убытки.
89 ● Цех работает эффективно! Уровень занятости: HIGH
90
91 <<< Цех «Кузовной_ремонт» – несёт убытки!
92 ● Следует увеличить кол-во механиков на: 1
93   ↳ Кол-во механиков станет: 3
94   ↳ Необслуженных заявок: 0
95
96 <<< Цех «Ремонт_двигателя» – не несёт убытки.
97 ● Цех работает эффективно! Уровень занятости: HIGH
98
99 ~~~~~

```

Рисунок 3.9 – Записані дані рекомендацій, щодо менеджменту персоналу

ВИСНОВОК

В роботі був описаний процес розробки програмної моделі, по обслуговуванню автомобілів в цехах автосервісу. Розроблена модель, працює в автономному режимі, за певним сценарієм, тобто конфігурації вхідних даних (параметрів) та методів (дій), які будуть обробляти їх. Для більшої складності та отримання достовірних результатів, заявки, на обслуговування автомобілів, генеруються випадковим чином та можуть містити декілька послуг.

Під час виконання програми, виконується візуалізація поточної ситуації в автосервісі, в реальному часі. Після закінчення, відбувається збір та підрахунок даних, та надаються статистика і рекомендації. Статистика, містить: загальну кількість отриманих заявок, вдало та невдало наданих послуг різного виду; середній час обслуговування одного автомобіля; середню довжину черг в кожному цеху; середню зайнятість робітників і їхню заробітну плату, загальний прибуток та збитки автосервісу. Рекомендації, містять поради, щодо менеджменту персоналу.

В першому розділі описується область визначення прикладного рішення та були розкриті ціль, актуальність та новизна роботи. Також, була розглянута поставлена задача предметної області.

У другому розділі було розглянуто методи візуалізації та реалізації прикладної задачі. Для моделювання було описано використання уніфікованої мови моделювання, а для реалізації прикладного рішення, розглянуті різноманітні інструментальні засоби.

Третій розділ описує, які саме інструментальні засоби були використані в процесі розробки програмної моделі. Також розглянуті побудова та тестування програмного продукту.

Виконані критерії, до програмного рішення, дають змогу з впевненістю сказати, що дане програмне забезпечення, особливо буде корисне, керівникам фірм (які спеціалізуються на обслуговуванні автомобілів).

СПИСОК ДЖЕРЕЛ ІНФОРМАЦІЇ

- 1 Kishori S. Beginning Java 8 Fundamentals. Language Syntax, Arrays, Data Types, Objects, and Regular Expressions. / Sharan Kishori., 2014. – 789 с. – (Apress).
- 2 Weisfeld M. The Object-Oriented Thought Process / Matt Weisfeld. // Addison-Wesely. – 2013. – №4. – С. 306.
- 3 Oracle. The Java Tutorials [Електронний ресурс] / Oracle – Режим доступу до ресурсу: <https://docs.oracle.com/javase/tutorial/java/index.html>.
- 4 Object Oriented Programming – Java OOPs Concepts With Examples [Електронний ресурс] – Режим доступу до ресурсу: <https://edureka.co/blog/object-oriented-programming>.
- 5 Fowler M. UML Distilled. A Brief Guide to the Standard Object Modeling Language / Martin Fowler. // Addison-Wesely. – 2004. – №3. – С. 175.
- 6 Larman C. Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development. / Creig Larman., 2004. – 627 с. – (Addison-Wesely).
- 7 Бублик В. В. Об'єктно-орієнтоване програмування [Електронний ресурс] / В. В. Бублик // ІТкнига. – 2015. – Режим доступу до ресурсу: http://itknyga.com.ua/docs/OOP_final.pdf.
- 8 Жуковський С. С. Об'єктно-орієнтоване програмування мовою C++ [Електронний ресурс] / С. С. Жуковський, Т. А. Вакалюк // ЖДУ. – 2016. – Режим доступу до ресурсу: http://eprints.zu.edu.ua/22367/1/%D0%BF%D0%BE%D1%81%D1%96%D0%B1%D0%BD0%B8%D0%BA%D0%A1_OOP%D1%81%D0%B0%D0%B9%D1%82.PDF.
- 9 Объектно-ориентированное программирование [Електронний ресурс] / Т. Бадд – Режим доступу до ресурсу: [http://khizha.dp.ua/library/Timothy_Budd - Introduction to OOP \(ru\).pdf](http://khizha.dp.ua/library/Timothy_Budd_-_Introduction_to_OOP_(ru).pdf).