

МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» им.В.И.УЛЬЯНОВА (ЛЕНИНА)

Кафедра вычислительной техники

Пояснительная записка к курсовой работе
по дисциплине «Объектно-ориентированное программирование»

Выполнил студент гр. XXXX _____ Name N. N

Проверил _____ Name N. N.

Санкт-Петербург

2021

Техническое задание

Введение

Разработать ПК (программный комплекс) для администратора салона красоты. В ПК должна храниться информация о клиентах салона, мастерах с указанием их специализации, услугах и их ценах. Администратор может добавлять, изменять или удалять эту информацию.

Цель и задачи

Целью курсового проектирования является закрепление и углубление теоретических знаний, приобретение практических навыков по проектированию и разработке программного обеспечения на объектно-ориентированном языке Java.

В задачи курсового проектирования входят:

- изучение особенностей конкретной предметной области, относящейся к заданию на курсовой проект, и разработка технического задания на программный комплекс (ПК);
- объектно-ориентированное проектирование ПК с использованием языка UML;
- разработка ПК на объектно-ориентированном языке;
- написание программной документации.

Основание для разработки

Основанием для разработки ПК – это курсовой проект по дисциплине «Объектно-ориентированное программирование».

Требования к программе

Требования к функциональным характеристикам

ПК должен обеспечивать выполнение следующих функций:

- просмотр, добавление, изменение базы данных (БД);
- выдача справочной информации, хранимой в БД.
- составление отчётов за день/неделю/месяц: расписание, доход, количество клиентов, лучших мастеров.

ПК должен обеспечивать ведение и хранение следующих данных:

- сведений о сотрудниках (мастерах) салона красоты;
- сведения о клиентах салона красоты;
- данные о проведённых услугах;
- сведения о проводимых в салоне красоты сервисах;
- каким сотрудникам салона красоты какие сервисы можно оказывать, а какие – нет;

Требования к организации и форме представления выходных данных

Выходные данные должны быть представлены в виде таблицы содержащей запрос пользователя.

Требования к организации и форме представления входных данных

Ввод исходных данных должен осуществляется в режиме диалога. Вводимые данные являются значениями характеристик редактируемых/добавляемых объектов.

Требования к надёжности

ПК должен устойчиво функционировать при соблюдении гарантии устойчивого функционирования ОС и СУБД. Должен быть обеспечен контроль входных данных на предмет соответствия предполагаемому типу.

Условия эксплуатации

Выполнение ПК своих функций должным образом будет соблюдаться при наличии стабильного подключения к СУБД.

Требование к информационной и программной совместимости

Выходная и входная информация ПК должна быть удобны для работы.

ПК должен быть выполнен на языке программирования Java и должен быть совместим с операционными системами GNU/Linux и Windows.

Обязательными требованиями при разработке кода ПК являются использование следующих конструкций языка Java:

- закрытые и открытые члены классов;
- наследование;
- конструкторы с параметрами;
- виртуальные функции;
- обработка исключительных ситуаций;
- динамическое создание объектов.

Требования к программной документации

Документация должна быть представлена в следующем составе:

- описание процесса проектирования ПК;
- руководство оператора;
- исходные тексты ПК.

Проектирование ПК

Описание вариантов использования ПК

Описание функциональных требований осуществляется на этапе проектирования комплекса. Для того чтобы детализировать требования, необходимо выделить процессы, происходящие в заданной предметной области. Описание таких процессов на UML выполняется в виде прецедентов (use case). Диаграмма прецедентов представлена на рисунке 1.

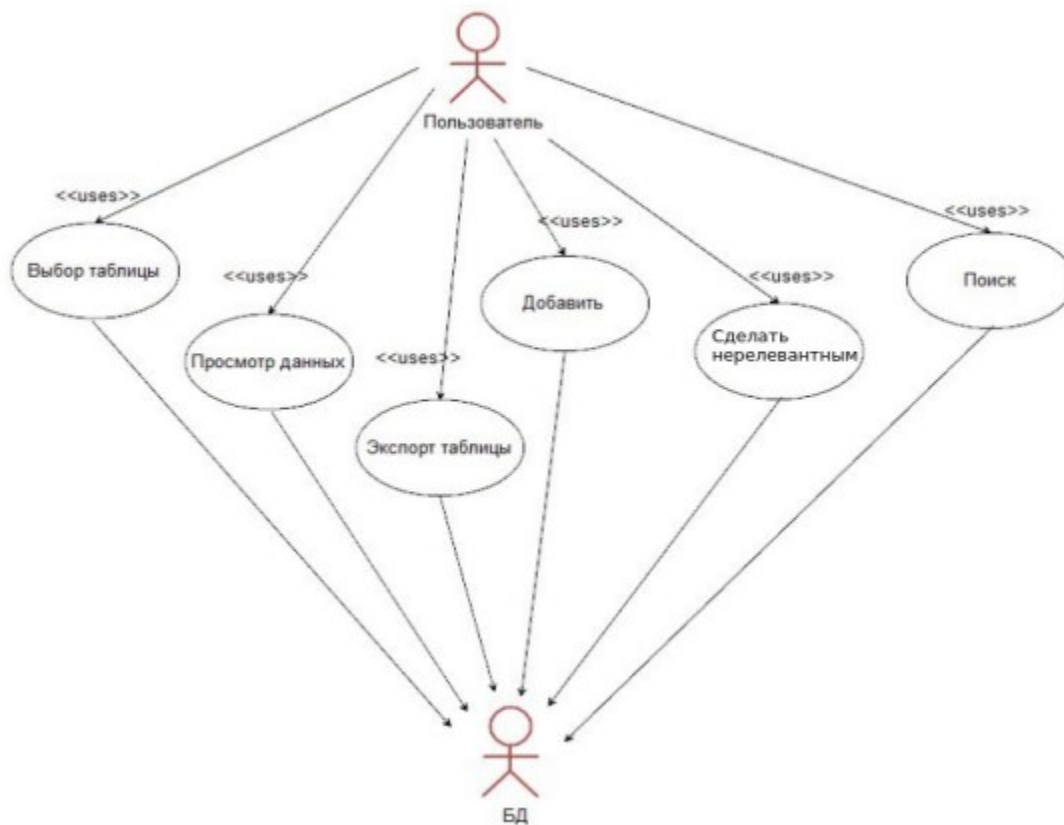


Рисунок 1. Диаграмма прецедентов

Создание прототипа интерфейса пользователя

Сначала необходимо описать экранные формы и элементы управления (кнопки, выпадающие списки, чекбоксы и др.). Такое описание представлено в таблице 1.

Таблица 1. Описание графических форм

Экранная форма	Элементы управления
Таблица сотрудников/клиентов/сервисов /типов сервисов	Поле поиска по производителям Кнопка «Загрузка таблицы» Кнопка «Сохранение» Кнопка «Добавление записи» Кнопка «Редактирования записи» Кнопка «Поиск»
Добавление сотрудников/клиентов/сервисов /типов сервисов	Поля для заполнения сведений Кнопка «Добавить»
Редактирование сотрудников/клиентов/сервисов /типов сервисов	Поля для редактирования сведений Кнопка «Добавить»
Фрейм задач	Поле с выбором дат Выпадающий список «Клиенты на запись» Выпадающий список «Сотрудники на запись» Выпадающий список «Сотрудники» Кнопка «Показать расписание» Кнопка «Показать нагрузку сотрудника» Кнопка «Показать запись клиента к сотруднику» Кнопка «Сформировать отчет PDF»

Примеры спроектированного пользовательского интерфейса представлены на рисунках 2, 3, 4, 5, 6, 7.

Choose table

ID	Service type	Date	Employee	Employee's cut	Client	Price	Relevance
195	Highlighting - 7	16.03.1972	Briggs Laurence - 9	945	Wagner Sommer - 65	1,176	true
196	Paint nails - 20	20.07.1982	Benton Lenny - 19	755	Logan Von - 115	1,244	true
197	Sugaring - 1	05.11.1992	Kennedy Mauro - 6	1,949	Hines Dinorah - 106	2,079	true
198	Depilation - 16	16.03.1992	Holden Geoffrey - 20	2,699	Hunter Mariano - 72	3,772	true
199	Massage - 4	17.04.1996	Benson Stanford - 18	1,981	Shepard Paris - 118	2,979	true
200	Pedicures - 6	01.10.2010	Potter ZulemaZub - 11	459	Schmidt Tula - 62	605	true
201	Depilation - 16	17.11.1960	Briggs Laurence - 9	1,511	Beck Mark - 39	3,009	true
202	Peeling - 11	21.04.1998	Benson Stanford - 18	1,696	Kelly Morris - 58	2,122	true
203	Braiding pigtails - 13	02.02.2011	Fitzgerald Jama - 15	2,639	Blankenship Ronald - 87	3,320	true
204	Mesotherapy - 19	20.03.1966	Giles Leonardo - 3	2,095	Sherman Ping - 29	2,112	true
206	Makeup - 5	25.05.2007	Cervantes Joel - 7	4,012	Gutierrez Zack - 111	4,539	true
207	Restore beauty - 21	22.01.1968	Snyder Fausto - 14	523	Moody Gabriele - 112	624	true
208	Pedicures - 6	26.07.1985	Kennedy Mauro - 6	1,773	Davenport Jefferey - 6	3,534	true
209	Massage - 4	27.10.2001	Parker Vincenzo - 10	2,770	Nichols Timika - 41	4,087	true
210	Mesotherapy - 19	05.02.1999	Holden Geoffrey - 20	1,865	Adkins Odelia - 144	2,733	true
211	Manicure - 10	07.04.1995	McLeod Antoine2 - 8	1,685	Kelly Morris - 58	2,210	true
212	Pedicures - 6	19.03.1976	Blankenship Brady - 21	1,095	Eaton Justin - 148	1,402	true
213	Ear piercing for childre...	30.01.1996	Hahn Moshe - 5	672	Knowles Walker - 56	1,278	true
214	Lawyer defense - 9	20.02.1960	Rosales Virgilio - 13	109	Cummings Pearly - 61	110	true
215	Haircut with blue lamin...	15.12.1978	Potter ZulemaZub - 11	2,963	Brady Kathline - 59	4,137	true
216	Braiding pigtails - 13	04.01.1957	Potter ZulemaZub - 11	3,080	Hunter Mariano - 72	4,842	true
217	Braiding pigtails - 13	04.09.1970	Snyder Fausto - 14	3,155	Mendoza Margaret - 15	3,865	true
218	MeowTherapy - 2	11.06.1999	Hahn Moshe - 5	2,584	Eaton Nonko - 60	4,595	true
219	Massage - 4	17.09.1963	Rosales Virgilio - 13	3,052	Guzman Tegan - 101	3,778	true
220	Restore beauty - 21	17.09.1956	Blankenship Brady - 21	2,409	Collier Julian - 55	2,891	true
221	Makeup - 5	14.11.1958	Benton Lenny - 19	2,003	Rivers Jarod - 137	3,932	true
222	Paint nails - 20	01.01.1980	Fitzgerald Jama - 15	2,480	Meadows Connie - 38	3,206	true
223	Sugaring - 1	21.02.1963	Blankenship Brady - 21	1,856	Knowles Walker - 56	3,197	true
224	Manicure - 10	27.12.1955	Alvarez Junko - 17	1,322	Tillman Darcey - 18	1,414	true
225	Wash hair - 12	13.11.2005	Benson Stanford - 18	3,200	Smith Aleisha - 14	3,573	true
226	Peeling - 11	19.01.1991	Hurst Kelly - 2	2,886	Flores Keith - 126	3,314	true
227	Highlighting - 7	23.12.1958	Potter ZulemaZub - 11	2,058	Molina Cletus - 35	2,065	true

Search:

☒ Considered

Рисунок 2. Пример формы с таблицами сервисов.

Choose table

ID	Full name	Gender	
23	Puka Kaka	M	123
18	Benson Stanford	M	6538 12
4	Haynes Haywood	M	0491 07
1	Dyer Annalee	F	3840 42
9	Briggs Laurence	M	1340 02
14	Snyder Fausto	F	6042 15
16	Vazquez Marquis	F	8483 95
21	Blankenship Brady	F	7181 80
22	Puka Adam	M	1235 25
11	Potter ZulemaZub	F	7996 02
15	Fitzgerald Jama	M	2427 75
6	Kennedy Mauro	F	9620 87
24	Lion Goose	M	2314 25
17	Alvarez Junko	F	6759 59
10	Parker Vincenzo	M	5476 39
8	McLeod Antoine2	M	2527 01
12	Moss Cleveland	F	3257 07
19	Benton Lenny	M	8587 35
5	Hahn Moshe	M	2600 34
2	Hurst Kelly	M	2761 64
20	Holden Geoffrey	F	3809 87
25	Watler Lion	M	1325 26
7	Cervantes Joel	F	5910 09
13	Rosales Virgilio	M	5201 54

Search:

Edit employee

ID is 21

First name:

Second name:

You selected: 16.02.2003

Day:

Month:

Year:

Passport:

Gender: ☐ Male ☒ Female

State: ☒ relevant

Permitted services:

Done!

Рисунок 3. Пример формы редактирования сотрудников.

Choose table

Frame of salon

Add specialization

What: Ear piercing for children - 14

Whom: Bennett Eladia - 4

Who: Potter ZulemaZub - 11

Client cost: 700.0

Employee's cut: 525.0

You selected: 05.05.2005

Date: Day: 5 Month: 5 Year: 2005

State: ☒ relevant

Done!

Message

Service (id=812) was saved in Data Base

OK

ID	Service type	Date	Employee	Employee's cut
136	Makeup - 5	12.06.1973	Snyder Fausto - 14	
137	MeowTherapy - 2	26.11.2004	Kennedy Mauro - 6	
138	Restore beauty - 21	18.04.1956	Hahn Moshe - 5	
139	Braiding pigtails - 13	23.06.2002	Alvarez Junko - 17	
140	Highlighting - 7	24.12.2002	Alvarez Junko - 17	
141	Manicure - 10	12.02.1984	Giles Leonardo - 3	
142	Nose cleaning - 18	16.03.1968	Haynes Haywood - 4	
143	Haircut with blue lamin...	16.12.1978	Fitzgerald Jama - 15	
144	Mesotherapy - 19	04.05.1969	Dyer Annalee - 1	
145	Wash hair - 12	18.03.1972		
146	Sugaring - 1	19.08.1967		
147	Mesotherapy - 19	07.08.1991		
148	Smear in the mud - 17	01.01.2003		
149	Smear in the mud - 17	11.09.1993		
150	Pedicures - 6	11.05.1970		
151	Ear piercing for childre...	28.04.1956		
152	Makeup - 5	08.11.1971	Haynes Haywood - 4	
153	Depilation - 16	20.05.1973	Briggs Laurence - 9	
154	Depilation - 16	25.06.1965	Potter ZulemaZub - 11	
155	Manicure - 10	06.10.1965	Potter ZulemaZub - 11	
156	Makeup - 5	24.05.1954	Vazquez Marquis - 16	
157	Smear in the mud - 17	22.05.1964	Haynes Haywood - 4	
158	MeowTherapy - 2	17.05.1994	Snyder Fausto - 14	
159	Depilation - 16	28.04.2004	Holden Geoffrey - 20	
160	Ear piercing for childre...	13.04.1991	Holden Geoffrey - 20	
161	Lawyer defense - 9	08.01.1983	Cervantes Joel - 7	
162	Smear in the mud - 17	29.01.1968	Hahn Moshe - 5	
163	Nose cleaning - 18	29.03.1982	Fitzgerald Jama - 15	
164	Restore beauty - 21	25.03.1980	Rosales Virgilio - 13	
165	Haircut with blue lamin...	15.02.1967	Moss Cleveland - 12	
166	Depilation - 16	06.08.1970	Haynes Haywood - 4	
167	Massage - 4	24.06.1996	Kennedy Mauro - 6	
168	Ear piercing for childre...	31.08.1995	Potter ZulemaZub - 11	
169	Contouring plastic - 8	14.07.1962	McLeod Antoine - 8	

Search:

Рисунок 4. Пример формы с добавление сервисов.

Task

Show schedule

Employee's workload

Specialist's appointment

Alvarez Junko - 17

Alvarez Junko - 17

Adkins Odelia - 144

Month

Week

Day

You selected: 15.12.2005

Day: 15 Month: 12 Year: 2005

ID	Service type	Date	Employee	Employee's cut	Client	Price	Relevance
599	Nail painting - 3	17.12.2005	Puka Kaka - 23	390.5	Sellers Macy - 117	500	true
734	Paint nails - 20	16.12.2005	Potter ZulemaZub - 11	425	O'Neill Tod - 113	500	true
599	Haircut with blue lami...	16.12.2005	Alvarez Junko - 17	10	Sellers Macy - 117	10	true
668	Test2 - 23	16.12.2005	Fitzgerald Jama - 15	0.99	Barnes Renetta - 66	1	true
722	Massage - 4	16.12.2005	Holden Geoffrey - 20	1,600	Vega Geoffrey - 142	2,000	true
574	Paint nails - 20	16.12.2005	Snyder Fausto - 14	425	Morrow Matt - 133	500	true
677	Test2 - 23	16.12.2005	Benson Stanford - 18	0.99	Moon Ricardo - 44	1	true
788	Depilation - 16	16.12.2005	Holden Geoffrey - 20	315	Cain Scot - 135	350	true
659	Paint nails - 20	16.12.2005	Lion Goose - 24	425	Ewing Carlita - 2	500	true
622	Highlighting - 7	16.12.2005	Alvarez Junko - 17	750	Dwa Aka - 151	1,000	true
515	Contouring plastic - 8	16.12.2005	Rosales Virgilio - 13	1,850	Silva Morgan - 77	2,500	true
755	Lawyer defense - 9	16.12.2005	Haynes Haywood - 4	9,900	Vinson Sandy - 94	10,000	true
547	Manicure - 10	16.12.2005	Cervantes Joel - 7	1,125	Parrish Marielle - 78	1,500	true
783	Peeling - 11	15.12.2005	Kennedy Mauro - 6	315	Stokes Gaynelle - 12	450	true
522	Lawyer defense - 9	15.12.2005	Hurst Kelly - 2	9,900	Douglas Oneida - 40	10,000	true
513	Highlighting - 7	15.12.2005	Potter ZulemaZub - 11	750	Copeland Dale - 50	1,000	true
751	Highlighting - 7	15.12.2005	Puka Adam - 22	750	Caldwell Jerald - 16	1,000	true
538	Manicure - 10	15.12.2005	Moss Cleveland - 12	1,125	Hopkins Orval - 26	1,500	true
549	MeowTherapy - 2	15.12.2005	Cervantes Joel - 7	0.5	Mcdonald Billie - 74	50	true
653	Ear piercing for childr...	15.12.2005	Haynes Haywood - 4	525	Carney Vania - 75	700	true
682	Test2 - 23	15.12.2005	Dyer Annalee - 1	0.99	Fry Yael - 90	1	true
562	Highlighting - 7	15.12.2005	Puka Adam - 22	750	Yates Randal - 45	1,000	true
721	Ear piercing for childr...	15.12.2005	Lion Goose - 24	525	Oliver Zada - 130	700	true
559	Makeup - 5	15.12.2005	Lion Goose - 24	350	Vega Geoffrey - 142	500	true
758	Contouring plastic - 8	15.12.2005	Haynes Haywood - 4	1,850	Oliver Zada - 130	2,500	true
754	Lawyer defense - 9	14.12.2005	Haynes Haywood - 4	9,900	Douglas Oneida - 40	10,000	true
768	Depilation - 16	14.12.2005	Snyder Fausto - 14	315	Molina Cletus - 35	350	true
592	Sugaring - 1	14.12.2005	Moss Cleveland - 12	212.5	Bates Juan - 109	250	true
517	Manicure - 10	14.12.2005	Hurst Kelly - 2	1,125	Shelton Eddy - 10	1,500	true
544	Nail painting - 3	14.12.2005	Cervantes Joel - 7	390.5	Owens Greg - 143	550	true
551	Makeup - 5	14.12.2005	Vazquez Marquis - 16	350	Herring Chia - 120	500	true
774	Pedicures - 6	14.12.2005	Vazquez Marquis - 16	1,101.57	Collier Julian - 55	1,509	true
542	Nail painting - 3	14.12.2005	Puka Kaka - 23	390.5	Aguilar Francesco - 28	550	true
607	Sugaring - 1	14.12.2005	Hurst Kelly - 2	212.5	Guthrie Cristin - 30	250	true

Income: 151816.0 Number of clients: 80 Best money: Moss Cleveland - 12
With percent: 27966.416400000002 Unique: 58 Best traffic: Alvarez Junko - 17

Рисунок 5. Пример формы задач.

ami...	16.12.2005	Alvarez Junko - 1 /	10/Sellers Macy - 11 /	10>true
	16.12.2005	Fitzgerald Jama - 15	0.99/Barnes Renetta - 66	1>true
	16.12.2005	Holden Geoffrey - 20	1,600/Vega Geoffrey - 142	2,000>true
	16.12.2005	Snyder Fausto - 14	425/Morrow Matt - 133	500>true
	16.12.2005	Benson Stanford - 18	0.99/Moon Ricardo - 44	1>true
	16.12.2005	Holden Geoffrey - 20	315/Cain Scot - 135	350>true
	16.12.2005	Lion Goose - 24	425/Ewinn Carlita - 2	500>true
	16.12.2005	Alvar		1,000>true
- 8	16.12.2005	Rosa		2,500>true
	16.12.2005	Hayn		10,000>true
	16.12.2005	Cerva		1,500>true
	15.12.2005	Kenn		450>true
	15.12.2005	Hurst		10,000>true
	15.12.2005	Potte		1,000>true
	15.12.2005	Puka		1,000>true
	15.12.2005	Moss		1,500>true
	15.12.2005	Cerva		50>true
ildr...	15.12.2005	Hayn		700>true
	15.12.2005	Dyer		1>true
	15.12.2005	Puka		1,000>true
ildr...	15.12.2005	Lion		700>true
	15.12.2005	Lion		500>true
- 8	15.12.2005	Hayn		2,500>true
	14.12.2005	Hayn		10,000>true
	14.12.2005	Snyd		350>true
	14.12.2005	Moss		250>true
	14.12.2005	Hurst		1,500>true
	14.12.2005	Cerva		550>true
	14.12.2005	Vazquez Marquis - 16	999/mering and - 120	500>true
	14.12.2005	Vazquez Marquis - 16	1,101.57/Collier Julian - 55	1,509>true
	14.12.2005	Puka Kaka - 23	390.5/Aguilar Francesco - 28	550>true
	14.12.2005	Hurst Kelly - 2	212.5/Guthrie Cristin - 30	250>true
	14.12.2005			
Income: 151816.0 Number of clients: 80 Best money: Moss Cleveland - 12				
With percent: 27066.416400000003 Unique: 59 Best traffic: Alvarez Junko - 17				

Рисунок 6. Пример выбора пути сохранения отчёта PDF.

Frame of salon

Choose table

ID	Full name	Gender	Passport	Birthday	Specializations	Active
10	Parker Vincenzo	M	5476 394814	09.02.2000	1: 7	true

Search: Vincenzo

☒ Considered

Рисунок 7. Пример поиска сотрудников.

Построение диаграммы программных классов

Диаграмма классов (class diagram) строится на основе объектной модели. В описание класса указываются три раздела: имя класса, состав компонентов класса и методы класса. Графически класс изображается в виде прямоугольника. Такая диаграмма представлена на рисунке 8.

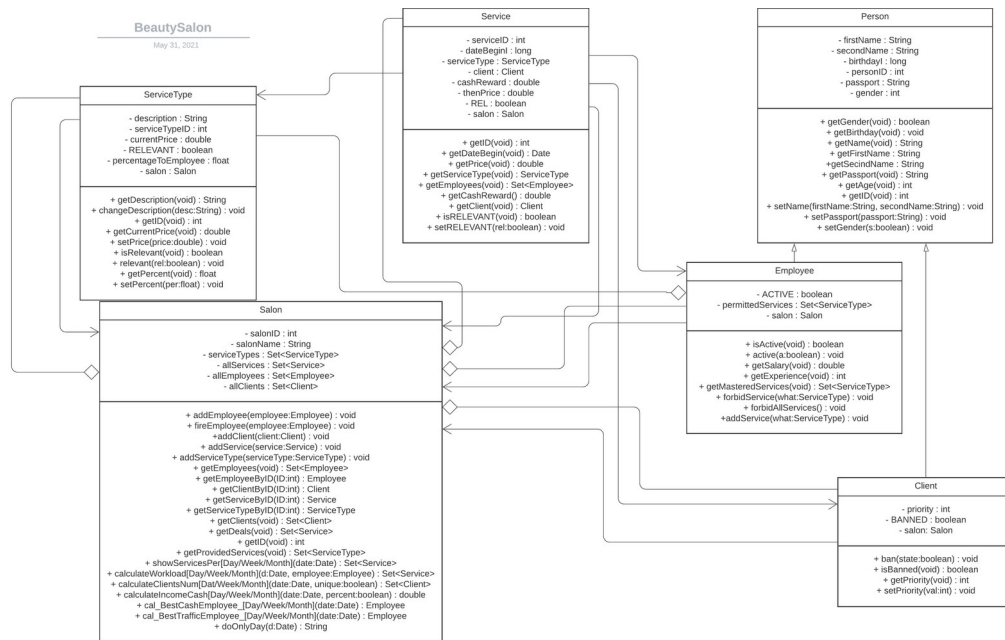


Рисунок 8. Диаграмма классов.

Описание классов

KY39.java

Класс, запускающий приложение.

Описание полей класса представлено в таблице 2.

Таблица 2 - описание полей класса KY39

Название	Тип данных	Семантика
private static final log	Logger	Логгер

Описание методов класса представлено в таблице 3.

Таблица 3 - описание методов класса KY39

Название метода	Возвращаемый тип	Описание параметров	Назначение
public static main	void	String[] args — аргументы программе	Главный метод main

Client.java

Класс, представляющий клиента.

Описание полей класса представлено в таблице 4.

Таблица 4 - описание полей класса Client

Название	Тип данных	Семантика
private priority	int	Приоритет клиента
private BANNED	boolean	Заблокирован ли клиент
private buffService	Service	Поле необходимо для установления связи OneToOne
private salon	Salon	Салон

Описание методов класса представлено в таблице 5.

Таблица 5 - описание методов класса Client

Название метода	Возвращаемый тип	Описание параметров	Назначение
public ban	void	boolean state – статус блокировки	Заблокировать клиента
public isBanned	boolean	–	Проверить заблокирован ли клиент
public getPriority	int	–	Получить приоритет клиента
public setPriority	void	int val – приоритет	Установить приоритет
public toString	String	–	Перевести объект в строку
public equals	boolean	Object otherObj – объект для сравнения	Сравнивает объекты

Employee.java

Класс, представляющий сотрудника салона красоты.

Описание полей класса представлено в таблице 6.

Таблица 6 - описание полей класса Employee

Название	Тип данных	Семантика
private ACTIVE	boolean	Активность сотрудника
private permittedServices	Set<ServiceType>	Разрешённые сервисы
private buffService	Service	Необходим для организации связи OneToOne
private salon	Salon	Салон

Описание методов класса представлено в таблице 7.

Таблица 7 - описание методов класса Employee

Название метода	Возвращаемый тип	Описание параметров	Назначение
public isActive	boolean		Проверить, активен ли сотрудник
public active	void	boolean a – активность сотрудника	Установить активность сотрудника
public synchronized getMasteredServices	Set<ServiceType>		Показывает в салоне услуги, которые сотрудник может оказывать
public synchronized addService	void	ServiceType what – услуга, которую может оказывать сотрудник	Если сотрудник компетентен в оказании услуги what, добавьте её к списку услуг, которые может оказывать сотрудник
public synchronized forbidService	void	ServiceType what – услуга, которую нельзя оказывать сотруднику	Запрещает сотруднику оказывать услугу what
public synchronized forbidAllServices	void		Запрещает сотруднику оказывать все услуги
public toString	String		Перевести объект в строку
public equals	boolean	Object otherObj – объект для сравнения	Сравнивает объекты

HibHundler.java

Класс организующий работу с hibernate.

Описание полей класса представлено в таблице 8.

Таблица 8 - описание полей класса HibHundler

Название	Тип данных	Семантика
private static factory	SessionFactory	Объект, генерирующий сессию
private static session	Session	Сессия

Описание методов класса представлено в таблице 9.

Таблица 9 - описание методов класса HibHundler

Название метода	Возвращаемый тип	Описание параметров	Назначение
public static loadSalon	Salon		Загрузить салон из БД
public static saveSalon	void	Salon salon – сохраняемый салон	Сохранить салон в БД
public static saveObject	void	Object obj – сущность	Сохранить сущность в БД
public static initFactoryAndSession	void		Проинициализировать factory и session для hibernate
public static closeFactoryAndSession	void		Закрыть factory и session для hibernate

InvalidNameException.java

Класс исключения при неправильном вводе имени.

Описание полей класса представлено в таблице 10.

Таблица 10 - описание полей класса InvalidNameException

Название	Тип данных	Семантика
kaka	String	Символ, из-за которого произошло исключение

Описание методов класса представлено в таблице 11.

Таблица 11 - описание методов класса InvalidNameException

Название метода	Возвращаемый тип	Описание параметров	Назначение
public getInvalidSymbol	String		Получить символ, который вызвал исключение

Person.java

Класс человек. От него наследуются Employee и Client.

Описание полей класса представлено в таблице 12.

Таблица 12 - описание полей класса Person

Название	Тип данных	Семантика
private personID	int	ID человека
private firstName	String	Имя человека
private secondName	String	Фамилия человека
private birthdayI	long	День рождения человека
private passport	String	Паспорт человека
private gender	int	Пол человека

Описание методов класса представлено в таблице 13.

Таблица 13 - описание методов класса Person

Название метода	Возвращаемый тип	Описание параметров	Назначение
public getID	int		Получить ID человека
public getName	String		Получить полное имя человека
public getFirstName	String		Получить только имя человека
public getSecondName	String		Получить фамилию человека
public setName	void	String firstName – имя String secondName – фамилия	Установить имя человека
public setPassport	void	String passport – паспорт человека	Установить паспорт
public setGender	void	boolean s – пол	Установить пол
public setBirthday	void	Date d – дата рождения	Задать дату рождения человека
public getBirthday	Date		Получить дату рождения
public getPassport	String		Получить паспорт человека
public getGender	boolean		Получить пол человека
public toString	String		Перевести объект в строку

ReportMaker.java

Класс, отвечающий за генерацию отчётов PDF.

Описание полей класса представлено в таблице 14.

Таблица 14 - описание полей класса ReportMaker

Название	Тип данных	Семантика
private static final log	Logger	Логгер
private FILE	String	Путь к файлу
private static t_font	Font	Шрифт параграфа
private static c_font	Font	Шрифт
private document	Document	Документ
private salon	Salon	Салон
private D_W_M	int	Дата, неделя или месяц
private d	Date	Дата для D_W_M
private ss	Set<Service>	Сервисы для отчёта

Описание методов класса представлено в таблице 15.

Таблица 15 - описание методов класса ReportMaker

Название метода	Возвращаемый тип	Описание параметров	Назначение
public synchronized static makeReport	void	Salon salon – Салон Set<Service> ss – Сервисы для отчёта int D_W_M – день=0/неделя=1/месяц=2 Date d – дата для D_W_M String path – путь к файлу	Сделать отчёт PDF
private addMetaData	void		Добавить метаданные для файла
private addTitle	void		Добавить заголовок
private dododo	void		Формирование таблицы и статистики
private takeDates	String		Получить дату-строку
private isOpen	boolean		Проверить открыт ли документ
private doOpen	void		Открыть документ
private doClose	void		Закрыть документ

Salon.java

Класс представляющий салон красоты.

Описание полей класса представлено в таблице 16.

Таблица 16 - описание полей класса Salon

Название	Тип данных	Семантика
private salonID	int	ID салона
private salonName	String	Название салона
private serviceTypes	Set<ServiceType>	Типы сервисов
private allServices	Set<Service>	Сервисы
private allEmployees	Set<Employee>	Сотрудники
private allClients	Set<Client>	Клиенты

Описание методов класса представлено в таблице 17.

Таблица 17 - описание методов класса Salon

Название метода	Возвращаемый тип	Описание параметров	Назначение
public synchronized addEmployee	void	Employee employee – сотрудник	Добавить сотрудника в салон красоты
public synchronized fireEmployee	void	Employee employee – сотрудник	Удалить сотрудника из салона красоты
public synchronized addClient	void	Client client – клиент	Добавить клиента
public synchronized addService	void	Service service – сервис	Добавить сервис
public synchronized addServiceType	void	ServiceType serviceType – тип сервиса	Добавить тип сервиса
public getName	String		Получить название салона красоты
public synchronized getEmployees	Set<Employee>		Получить сотрудников
public synchronized getClients	Set<Client>		Получить клиентов
public synchronized getProvidedServices	Set<ServiceType>		Получить сервисы
public synchronized getDeals	Set<Service>		Получить типы сервисов

public getID	int		Получить ID салона красоты
public synchronized showServicesPerDay	Set<Service >	Date day – день	Посмотреть все услуги, которые проводились в день day (укажите любое время этого дня)
public synchronized showServicesPerWeek	Set<Service >	Date week – неделя	Посмотреть все услуги, которые проводились в неделю week (укажите любое время этой недели)
public synchronized static getWeekKnowsDay	Date[]	Date oneDayOfThisWeek – день недели	Получить дни недели
public synchronized showServicesPerMonth	Set<Service >	Date month – месяц	Посмотреть все услуги, которые проводились в месяц month (укажите любое время этого месяца)
public calculateWorkloadMonth	Set<Service >	Date month – месяц Employee employee – сотрудник салона	Вычислить загруженность сотрудника за месяц month (укажите любую дату из этого месяца)
public calculateWorkloadWeek	Set<Service >	Date week – неделя Employee employee – сотрудник салона	Вычислить загруженность сотрудника за неделю week (укажите любой день интересующей недели)
public calculateWorkloadDay	Set<Service >	Date day – день Employee employee – сотрудник салона	Вычислить загруженность сотрудника за день day (укажите любое время интересующего дня)
public calculateClientRaspMonth	Set<Service >	Date month – месяц Employee employee – сотрудник салона Client client – клиент салона	Вычислить клиента client специалисту employee за месяц month (укажите любую дату из этого месяца)
public calculateClientRaspWeek	Set<Service >	Date week – неделя Employee employee – сотрудник салона Client client – клиент салона	Вычислить клиента client специалисту employee за неделю week (укажите любой

			день интересующей недели)
public calculateClientRaspDay	Set<Service >	Date day – день Employee employee – сотрудник салона Client client – клиент салона	Вычислить клиента client специалисту employee за день day (укажите любое время интересующего дня)
public calculateClientsNumMo nth	int	Date month – месяц boolean unique – Если unique == true, то покажет только уникальных клиентов	Посчитать клиентов за месяц (укажите любую дату интересующего месяца)
public calculateClientsNumWe ek	int	Date week – неделя boolean unique – Если unique == true, то покажет только уникальных клиентов	Посчитать клиентов за неделю (укажите любой день интересующей недели)
public calculateClientsNumDa y	int	Date day – день boolean unique – Если unique == true, то покажет только уникальных клиентов	Посчитать клиентов за день
public calculateIncomeCashM onth	double	Date month – месяц boolean percent – Если percent == true, то вычислит доход с учётом выплаченных процентов сотрудникам, иначе полную сумму	Вычислить приход денег за месяц (укажите любую дату интересующего месяца)
public calculateIncomeCashWe ek	double	Date week – неделя boolean percent – Если percent == true, то вычислит доход с учётом выплаченных процентов сотрудникам, иначе полную сумму	Вычислить приход денег за неделю (укажите любой день интересующей недели)
public calculateIncomeCashDa y	double	Date day – день boolean percent – Если percent == true, то вычислит доход с учётом выплаченных процентов сотрудникам, иначе полную сумму	Вычислить приход денег за день
public cal_BestCashEmployee _Month	Employee	Date month – месяц	Вычислить лучшего по прибыли работника за месяц (укажите любую дату интересующего месяца)
public cal_BestCashEmployee _Week	Employee	Date week – неделя	Вычислить лучшего по прибыли работника за неделю (укажите любой день интересующей недели)

public cal_BestCashEmployee _Day	Employee	Date day – день	Вычислить лучшего по прибыли работника за день
public cal_BestTrafficEmployee _Month	Employee	Date month – месяц	Вычислить лучшего по посещаемости работника за месяц (укажите любую дату интересующего месяца)
public cal_BestTrafficEmployee _Week	Employee	Date week – неделя	Вычислить лучшего по посещаемости работника за неделю (укажите любой день интересующей недели)
public cal_BestTrafficEmployee _Day	Employee	Date day – день	Вычислить лучшего по посещаемости работника за день
public toString	String		Перевести объект в строку
public synchronized getEmployeeByID	Employee	int ID – ID сотрудника	Получить сотрудника по его ID
public synchronized getClientByID	Client	int ID – ID клиента	Получить клиента по его ID
public synchronized getServiceByID	Service	int ID – ID сервиса	Получить сервис по его ID
public synchronized getServiceTypeByID	ServiceType	int ID – ID типа сервиса	Получить тип сервиса по его ID
public static doOnlyDate	String	Date d – дата для преобразования	Преобразует дату в красиво читаемый вид

Service.java

Класс, отражающий сервис. Запись о том, какой сотрудник какому клиенту какую оказывал услугу, перечисленную в `ServiceType`, и за сколько.

Описание полей класса представлено в таблице 18.

Таблица 18 - описание полей класса `Service`

Название	Тип данных	Семантика
<code>private serviceID</code>	<code>int</code>	ID сервиса
<code>private dateBeginI</code>	<code>long</code>	Дата начала проведения
<code>private cashReward</code>	<code>double</code>	Вознаграждение сотруднику
<code>private thenPrice</code>	<code>double</code>	Сколько заплатил клиент
<code>private REL</code>	<code>boolean</code>	Актуальность записи
<code>private serviceType</code>	<code>ServiceType</code>	Тип сервиса
<code>private client</code>	<code>Client</code>	Клиент, которому оказывали услугу
<code>private whoDo</code>	<code>Employee</code>	Сотрудник, который оказывал услуга
<code>private salon</code>	<code>Salon</code>	Салон, где оказывали услугу

Описание методов класса представлено в таблице 19.

Таблица 19 - описание методов класса `Service`

Название метода	Возвращаемый тип	Описание параметров	Назначение
<code>public getID</code>	<code>int</code>		Получить ID сервиса
<code>public getDateBegin</code>	<code>Date</code>		Получить дату начала сервиса
<code>public getPrice</code>	<code>double</code>		Получить сумму, которую заплатил клиент
<code>public getServiceType</code>	<code>ServiceType</code>		Получить тип сервиса
<code>public getEmployee</code>	<code>Employee</code>		Получить работника, который оказывал услугу
<code>public getCashReward</code>	<code>double</code>		Возвращает сумму, которую получил сотрудник после оказания услуги

public getClient	Client		Получить клиента, которому оказывали услугу
public isRELEVANT	boolean		Просмотреть актуальность записи
public setRELEVANT	void	boolean rel – актуальность записи	Возможно эта запись создана с ошибкой или ещё какая-нибудь причина, по которой она не должна считаться
public toString	String		Перевести объект в строку

ServiceType.java

Класс представляющий тип сервиса.

Описание полей класса представлено в таблице 20.

Таблица 20 - описание полей класса ServiceType

Название	Тип данных	Семантика
private serviceTypeID	int	ID типа сервиса
private description	String	Описание
private currentPrice	double	Цена
private percentageToEmployee	float	Проценты сотруднику
private RELEVANT	boolean	Актуальность
private whoMastered	Set<Employee>	Сотрудники, которые могут оказывать данную услугу
private buffService	Service	Поле необходимо для организации связи OneToOne
private salon	Salon	Салон

Описание методов класса представлено в таблице 21.

Таблица 21 - описание методов класса ServiceType

Название метода	Возвращаемый тип	Описание параметров	Назначение
public getDescription	String		Получить описание типа сервиса
public synchronized getWhoMasteredThisServiceTypes	Set<Employee>		Получить список сотрудников, которые могут оказывать данный сервис
public changeDescription	void	String desc – новое описание	Поменять описание типа сервиса
public getID	int		Получить ID сервиса
public getCurrentPrice	double		Получить текущую стоимость услуги
public setPrice	void	double price – новая цена	Установить цену типа сервиса
public isRelevant	boolean		Проверить, актуален ли данный тип сервиса
public relevant	void	boolean rel – актуальность	Установить актуальность сервиса

public getPercent	float		Посмотреть процент, который начисляется сотруднику, от суммы стоимости услуги
public setPercent	void	float employeePercent – процент	Установить процент, который начисляется сотруднику, от суммы стоимости услуги
public addServiceTypeMaster	void	Employee who – сотрудник	Добавить сотрудника, который может оказывать данный сервис
public removeServiceTypeMaster	void	Employee who – сотрудник	Убрать сотрудника, который раньше мог оказывать данный сервис
public toString	String		Перевести объект в строку

AddClient.java

Форма для добавления клиента.

Описание полей класса представлено в таблице 22.

Таблица 22 - описание полей класса AddClient

Название	Тип данных	Семантика
private static final log	Logger	Логгер
salonGUI	SalonGUI	Главное окно приложения
salon	Salon	Салон
private final DEFAULT_WIDTH	int	Стандартная ширина
private final DEFAULT_HEIGHT	int	Стандартная высота
private mainPanel	JPanel	Главная панель
private doneButton	JButton	Кнопка завершения добавления
private firstNameText	JTextField	Текстовое поля для имени
private secondNameText	JTextField	Текстовое поле для фамилии
private dateChooser	DateChooser	Панель выбора даты
private passportText	JTextField	Текстовое поле для паспорта
private gender	boolean	Пол
private state	boolean	Состояние
private priorityText	JTextField	Текстовое поле для приоритета

Описание методов класса представлено в таблице 23.

Таблица 23 - описание методов класса AddClient

Название метода	Возвращаемый тип	Описание параметров	Назначение
public updateAdd	void		Обновить содержимое формы
public getPreferredSize	Dimension		Определить предпочитаемый размер
private addClient	void		Добавить клиента в салон

AddEmployee.java

Форма для добавления сотрудника.

Описание полей класса представлено в таблице 24.

Таблица 24 - описание полей класса AddEmployee

Название	Тип данных	Семантика
private static final log	Logger	Логгер
salonGUI	SalonGUI	Салон
salon	Salon	Салон
private final DEFAULT_WIDTH	int	Стандартная ширина
private final DEFAULT_HEIGHT	int	Стандартная высота
private mainPanel	JPanel	Главная панель
private doneButton	JButton	Кнопка завершения добавления
private firstNameText	JTextField	Текстовое поля для имени
private secondNameText	JTextField	Текстовое поле для фамилии
private dateChooser	DateChooser	Панель выбора даты
private passportText	JTextField	Текстовое поле для паспорта
private gender	boolean	Пол
private state	boolean	Состояние
private permittedServicesText	JTextField	Текстовое поле для разрешённых сервисов

Описание методов класса представлено в таблице 25.

Таблица 25 - описание методов класса AddEmployee

Название метода	Возвращаемый тип	Описание параметров	Назначение
public updateAdd	void		Обновить содержимое формы
public getPreferredSize	Dimension		Определить предпочитаемый размер
private addEmployee	void		Добавить сотрудника в салон

AddService.java

Форма для добавления сервиса.

Описание полей класса представлено в таблице 26.

Таблица 26 - описание полей класса AddService

Название	Тип данных	Семантика
private static final log	Logger	Логгер
salonGUI	SalonGUI	Главное окно приложения
salon	Salon	Салон
private final DEFAULT_WIDTH	int	Стандартная ширина
private final DEFAULT_HEIGHT	int	Стандартная высота
private mainPanel	JPanel	Главная панель
private doneButton	JButton	Кнопка завершения добавления
private clientCostText	JTextField	Текстовое поля для стоимости
private employeeSalaryText	JTextField	Текстовое поля для доли сотруднику
private dateChooser	DateChooser	Панель выбора даты
private selectedServiceType	int	Номер выбранного типа сервиса
private selectedEmployee	int	Номер выбранного сотрудника
private selectedClient	int	Номер выбранного клиента
private state	boolean	Состояние

Описание методов класса представлено в таблице 27.

Таблица 27 - описание методов класса AddService

Название метода	Возвращаемый тип	Описание параметров	Назначение
public updateAdd	void		Обновить содержимое формы
private static takeIDfromCombo	int	String s – элемент выпадающего списка	Парсер ID
private setEmployeeSalaryText	void		Установить
public getPreferredSize	Dimension		Определить предпочитаемый размер
private addService	void		Добавить сервис в салон

AddServiceType.java

Форма для добавления типа сервиса.

Описание полей класса представлено в таблице 28.

Таблица 28 - описание полей класса AddServiceType

Название	Тип данных	Семантика
private static final log	Logger	Логгер
salonGUI	SalonGUI	Главное окно приложения
salon	Salon	Салон
private final DEFAULT_WIDTH	int	Стандартная ширина
private final DEFAULT_HEIGHT	int	Стандартная высота
private mainPanel	JPanel	Главная панель
private doneButton	JButton	Кнопка завершения добавления
private descriptionText	JTextField	Текстовое поля для описания
private currentPriceText	JTextField	Текстовое поля для цены
private percentageToEmployeeText	JTextField	Текстовое поля для процента
private state	boolean	Состояние

Описание методов класса представлено в таблице 29.

Таблица 29 - описание методов класса AddServiceType

Название метода	Возвращаемый тип	Описание параметров	Назначение
public updateAdd	void		Обновить содержимое формы
public addPreferredSize	Dimension		Определить предпочитаемый размер
private addServiceType	void		Добавить тип сервиса в салон

DateChooser.java

Панель выбора даты.

Описание полей класса представлено в таблице 30.

Таблица 30 - описание полей класса DateChooser

Название	Тип данных	Семантика
private yComboStr	String[]	Массив годов
private mComboStr	String[]	Массив месяцев
private dComboStr	String[]	Массив дней
private yCombo	JComboBox	Выпадающий список годов
private mCombo	JComboBox	Выпадающий список месяцев
private dCombo	JComboBox	Выпадающий список дней
private textDate	JTextField	Текстовое поле для даты
private selectedYear	int	Выбранный год
private selectedMonth	int	Выбранный месяц
private selectedDay	int	Выбранный день

Описание методов класса представлено в таблице 31.

Таблица 31 - описание методов класса DateChooser

Название метода	Возвращаемый тип	Описание параметров	Назначение
public getDate	Date		Получить выбранную дату
public setDate	void	Date date – дата	Установить дату
private get_yComboStr	String[]		Получить массив лет
private get_mComboStr	String[]		Получить массив месяцев
private get_dComboStr	String[]		Получить массив дней
public static doStringDate	String	Date d – дата	Перевод даты в строку

EditClient.java

Форма для редактирования клиента.

Описание полей класса представлено в таблице 32.

Таблица 32 - описание полей класса EditClient

Название	Тип данных	Семантика
private static final log	Logger	Логгер
salonGUI	SalonGUI	Главное окно приложения
salon	Salon	Салон
private final DEFAULT_WIDTH	int	Стандартная ширина
private final DEFAULT_HEIGHT	int	Стандартная высота
private mainPanel	JPanel	Главная панель
private doneButton	JButton	Кнопка завершения добавления
private firstNameText	JTextField	Текстовое поля для имени
private secondNameText	JTextField	Текстовое поле для фамилии
private dateChooser	DateChooser	Панель выбора даты
private gender	boolean	Пол
private state	boolean	Состояние
private priorityText	JTextField	Текстовое поле для приоритета

Описание методов класса представлено в таблице 33.

Таблица 33 - описание методов класса EditClient

Название метода	Возвращаемый тип	Описание параметров	Назначение
public updateEdit	void	Client c – клиент	Обновить содержимое формы
public getPreferredSize	Dimension		Определить предпочитаемый размер
private changeClient	void	Client c – клиент	Отредактировать клиента

EditEmployee.java

Форма для редактирования сотрудника.

Описание полей класса представлено в таблице 34.

Таблица 34 - описание полей класса EditEmployee

Название	Тип данных	Семантика
private static final log	Logger	Логгер
salonGUI	SalonGUI	Главное окно приложения
salon	Salon	Салон
private final DEFAULT_WIDTH	int	Стандартная ширина
private final DEFAULT_HEIGHT	int	Стандартная высота
private mainPanel	JPanel	Главная панель
private doneButton	JButton	Кнопка завершения добавления
private firstNameText	JTextField	Текстовое поля для имени
private secondNameText	JTextField	Текстовое поля для фамилии
private dateChooser	DateChooser	Панель выбора даты
private passportText	JTextField	Текстовое поля для паспорта
private gender	boolean	Пол
private state	boolean	Состояние
private permittedServicesText	JTextField	Текстовое поля для разрешённых сервисов

Описание методов класса представлено в таблице 35.

Таблица 35 - описание методов класса EditEmployee

Название метода	Возвращаемый тип	Описание параметров	Назначение
public updateEdit	void	Employee e – сотрудник	Обновить содержимое формы
public getPreferredSize	Dimension		Определить предпочитаемый размер
private changeEmployee	void	Employee e – сотрудник	Отредактировать сотрудника

EditService.java

Форма для редактирования сервиса.

Описание полей класса представлено в таблице 36.

Таблица 36 - описание полей класса EditService

Название	Тип данных	Семантика
private static final log	Logger	Логгер
salonGUI	SalonGUI	Главное окно приложения
salon	Salon	Салон
private final DEFAULT_WIDTH	int	Стандартная ширина
private final DEFAULT_HEIGHT	int	Стандартная высота
private mainPanel	JPanel	Главная панель
private doneButton	JButton	Кнопка завершения добавления
private state	boolean	Состояние

Описание методов класса представлено в таблице 37.

Таблица 37 - описание методов класса EditService

Название метода	Возвращаемый тип	Описание параметров	Назначение
public updateEdit	void	Service s – сервис	Обновить содержимое формы
public getPreferredSize	Dimension		Определить предпочитаемый размер
private changeService	void	Service s – сервис	Отредактировать сервис

EditServiceType.java

Форма для редактирования тип сервиса.

Описание полей класса представлено в таблице 38.

Таблица 38 - описание полей класса EditServiceType

Название	Тип данных	Семантика
private static final log	Logger	Логгер
salonGUI	SalonGUI	Главное окно приложения
salon	Salon	Салон
private final DEFAULT_WIDTH	int	Стандартная ширина
private final DEFAULT_HEIGHT	int	Стандартная высота
private mainPanel	JPanel	Главная панель
private doneButton	JButton	Кнопка завершения добавления
private descriptionText	JTextField	Текстовое поля для описания
private currentPriceText	JTextField	Текстовое поля для цены
private percentageToEmployeeText	JTextField	Текстовое поля для процента сотруднику
private state	boolean	Состояние

Описание методов класса представлено в таблице 39.

Таблица 39 - описание методов класса EditServiceType

Название метода	Возвращаемый тип	Описание параметров	Назначение
public updateEdit	void	ServiceType st – тип сервиса	Обновить содержимое формы
public getPreferredSize	Dimension		Определить предпочитаемый размер
private changeServiceType	void	ServiceType st – тип сервиса	Отредактировать тип сервиса

LoadingMessage.java

Фрейм загрузки.

Описание полей класса представлено в таблице 40.

Таблица 40 - описание полей класса LoadingMessage

Название	Тип данных	Семантика
private final DEFAULT_WIDTH	int	Стандартная ширина
private final DEFAULT_HEIGHT	int	Стандартная высота
private picPanel	JPanel	Панель с анимацией
private imgs	ArrayList<JLabel>	Список кадров
private msg	String	Сообщение
private msgLabel	JLabel	Текстовая пометка с сообщением
private ALIVE	boolean	Идёт ли анимация

Описание методов класса представлено в таблице 41.

Таблица 41 - описание методов класса LoadingMessage

Название метода	Возвращаемый тип	Описание параметров	Назначение
public getPreferredSize	Dimension		Определить предпочитаемый размер
public showLoad	void	boolean SHOWED – показать ли панель	Показать панель

SalonGUI.java

Главное окно приложения.

Описание полей класса представлено в таблице 42.

Таблица 42 - описание полей класса SalonGUI

Название	Тип данных	Семантика
private static salonGUI	SalonGUI	Главное окно приложения
private salon	Salon	Салон
private static employeesCol	String[]	Колонка таблицы для сотрудников
private static clientsCol	String[]	Колонка таблицы для клиентов
private static servicesCol	String[]	Колонка таблицы для сервисов
private static serviceTypesCol	String[]	Колонка таблицы для типов сервисов
private DBLOADED	boolean	Флаг загрузки БД
private mainFrame	JFrame	Главный фрейм
private saveButton	JButton	Кнопка сохранения в БД
private openButton	JButton	Кнопка загрузки из БД
private addButton	JButton	Кнопка добавления записи
private editButton	JButton	Кнопка редактирования записи
private taskButton;	JButton	Кнопка открытия окна задач
private bottomPanel	JPanel	Нижняя панель
private bottomLabel	JLabel	Метка для поля поиска
private bottomText	JTextField	Текстовое поле для текста поиска
private bottomButton	JButton	Кнопка найти
private consideredCheckBox	JCheckBox	Флаг рассмотрения скрытых записей
private soaedToolBar	JToolBar	Панель инструментов с кнопками
private currentTable	JPanel	Панель с текущей таблицей
private employeeModel	DefaultTableModel	Модель для таблицы для сотрудников
private employeeTable	JTable	Таблица для сотрудников
private employeePane	JScrollPane	Панель с ползунком для сотрудников
private clientModel	DefaultTableModel	Модель для таблицы для клиентов
private clientTable	JTable	Таблица для клиентов
private clientPane	JScrollPane	Панель с ползунком для клиентов
private serviceModel	DefaultTableModel	Модель для таблицы для сервисов
private serviceTable	JTable	Таблица для сервисов

private servicePane	JScrollPane	Панель с ползунком для сервисов
private serviceTableModel	DefaultTableModel	Модель для таблицы для типов сервисов
private serviceTypeTable	JTable	Таблица для типов сервисов
private serviceTypePane	JScrollPane	Панель с ползунком для типов сервисов
private chooseTableMenuBar	JMenuBar	Меню бар
private chooseTableMenu	JMenu	Элемент меню бара
private employeeEditor	EditEmployee	Форма для редактирования сотрудников
private clientEditor	EditClient	Форма для редактирования клиентов
private serviceEditor	EditService	Форма для редактирования сервисов
private serviceTypeEditor	EditServiceType	Форма для редактирования типов сервисов
private employeeAdder	AddEmployee	Форма для добавления сотрудника
private clientAdder	AddClient	Форма для добавления клиента
private serviceAdder	AddService	Форма для добавления сервиса
private serviceTypeAdder	AddServiceType	Форма для добавления типа сервиса
private taskHundler	TaskHundler	Панель задач
private curPane	JScrollPane	Текущая выбранная панель
private filter	String	Фильтр
private filters	String[]	Слова для фильтрации
private CONSIDERED	boolean	Флаг рассмотрения скрытых записей
private loadLoading	LoadingMessage	Фрейм загрузки для загрузки из БД
private saveLoading	LoadingMessage	Фрейм загрузки для сохранения в БД

Описание методов класса представлено в таблице 43.

Таблица 43 - описание методов класса SalonGUI

Название метода	Возвращаемый тип	Описание параметров	Назначение
public static start	void		Запуск приложения
public show	void		Показать главное окно
private PrapareTables	void		Подготовить таблицы
private checkPrepare	boolean		Проверить готовность
private initTablesOf_Employees	void		Инициализировать таблицу сотрудников
public refreshTablesOf_Employees	void		Обновить таблицу сотрудников

private getEmployees_DefaultTableModel	DefaultTableModel		Получить модель для таблицы сотрудников
private initTablesOf_Clients	void		Инициализировать таблицу клиентов
public refreshTablesOf_Clients	void		Обновить таблицу клиентов
private getClients_DefaultTableModel	DefaultTableModel		Получить модель для таблицы клиентов
private initTablesOf_Services	void		Инициализировать таблицу сервисов
public refreshTablesOf_Service	void		Обновить таблицу сервисов
private getServices_DefaultTableModel	DefaultTableModel		Получить модель для таблицы сервисов
private initTablesOf_ServiceTypes	void		Инициализировать таблицу типов сервисов
public refreshTablesOf_ServiceTypes	void		Обновить таблицу типов сервисов
private getServiceTypes_DefaultTableModel	DefaultTableModel		Получить модель для таблицы типов сервисов
private refresh	void	JScrollPane cur – текущая выбранная панель	Обновить всё
private refreshTables	void		Обновить таблицы
private loadSalon	void		Загрузить салон из БД
private saveSalon	void		Сохранить сало в БД
private removeVisibleWindows	void		Закрыть открытые окна
private employees2Table	Object[][]	Set<Employee> emps – сотрудники	Перевести сотрудников в строки таблицы
private clients2Table	Object[][]	Set<Client> clients – клиенты	Перевести клиентов в строки таблицы
private services2Table	Object[][]	Set<Service> ss - сервисы	Перевести сервисы в строки таблицы
private serviceTypes2Table	Object[][]	Set<ServiceType> sts – типы сервисов	Перевести типы сервисов в строки таблицы

public getSalon	Salon		Получить салон
public static getSalonGUI	SalonGUI		Получить главное окно приложения
private filtering	boolean	Object obj – запись	Профилировать запись
private checkFilterAndProperties	boolean	String prop – поля объекта String[] filters – слова для фильтрации	Проверить проходит ли запись через фильтр
private static makeEntityStringProperties	String	Object obj – запись	Получить строку полей объекта записи
private editItem	void	JScrollPane cur – текущая выбранная панель	Отредактировать запись
private addItem	void	JScrollPane cur – текущая выбранная панель	Добавить запись
private doTask	void		Показать окно задач

TaskHundler.java

Панель задач.

Описание полей класса представлено в таблице 44.

Таблица 44 - описание полей класса TaskHundler

Название	Тип данных	Семантика
private static final log	Logger	Логгер
salonGUI	SalonGUI	Главное окно приложения
salon	Salon	Салон
private mainPanel	JPanel	Главная панель
private tablePanel	JPanel	Панель с таблицей
private upPanel	JPanel	Верхняя панель
private downPanel	JPanel	Нижняя панель
private dc	DateChooser	Панель выбора даты
private choosedEmployeeWorkload	Employee	Выбранный сотрудник для показа его загрузки
private choosedEmployeeRecord	Employee	Выбранный сотрудник для показа записей к нему
private choosedClientRecord	Client	Выбранный клиент для показа записей
private REPORTING	boolean	Делается ли сейчас отчёт
lastChoosedServices	Set<Service>	Последние выбранные сервисы
private D_W_M	int	День/неделя/месяц

Описание методов класса представлено в таблице 45.

Таблица 45 - описание методов класса TaskHundler

Название метода	Возвращаемый тип	Описание параметров	Назначение
public update	void		Обновить содержимое формы
public getPreferredSize	Dimension		Получить предпочитаемый размер
private prepareRasp	void	JPanel upPanel – верхняя панель	Подготовить панель расписанием работы салона
private prepareEmployeeRasp	void	JPanel upPanel – верхняя панель	Подготовить панель с расписанием сотрудника
private prepareClientRasp		JPanel upPanel – верхняя панель	Подготовить панель с записями клиентов к

			сотрудникам
private static prepareEmployeeList	String[]	Set<Employee> es – сотрудники	Перевести сотрудников в массив строк
private static prepareClientList	String[]	Set<Employee> cs – клиенты	Перевести клиентов в массив строк
private static takeIDfromCombo	int	String s – элемент выпадающего списка	Получить ID из элемента выпадающего списка
public prepare_D_W_M	void	JPanel upPanel – верхняя панель	Подготовить панель с выбором даты/недели/месяца
private preparateReportButton	void	JPanel upPanel – верхняя панель	Подготовить панель с генерацией отчёта PDF
public makeTable	void	Set<Service> services – сервисы для таблицы	Отобразить таблицу
private makeStatistics	void	JPanel downPanel – нижняя панель	Посчитать статистику
private services2Table	Object[][]	Set<Service> ss – сервисы для таблицы	Перевести сервисы в таблицу

Руководство оператору

Назначение программы

ПК «Администрирование сети салонов красоты» должен входить в состав автоматизированной системы учёта и администрирования информации.

В рамках ПК «Администрирование салона красоты» администратор может:

- добавлять, править и делать нерелевантной информацию о сотрудниках;
- добавлять, править и делать нерелевантной информацию о клиентах;
- добавлять, править и делать нерелевантной информацию о сервисах;
- добавлять, править и делать нерелевантной информацию о типах сервисах;
- получать отчёты о работе салона красоты;

Описание задачи

В ПК должны храниться сведения клиентах салона, мастерах с указанием их специализации, услугах и их ценах. Администратор сети магазинов может добавлять, изменять и удалять эту информацию.

Обязательными требованиями при разработке кода ПК являются использование следующих конструкций языка Java:

- закрытые и открытые члены классов;
- наследование;
- конструкторы с параметрами;
- абстрактные базовые классы;
- виртуальные функции;
- обработка исключительных ситуаций;
- динамическое создание объектов.

На основании требований были разработаны программные классы.

Требования к коду ПК учтены при создании программных классов и непосредственном написании программы.

Входные и выходные данные

Выходные данные должны быть представлены в виде таблицы содержащей необходимые данные

Ввод исходных данных должен осуществляется в режиме диалога. Вводимые данные являются значениями характеристик информационных объектов. Вводимая информация может выбираться или набираться из списка предлагаемых значений.

Запуск программы

При запуске программы на экране появится окно. Необходимо загрузить данные из БД. Соответствующая кнопка представлена на рисунке 9.

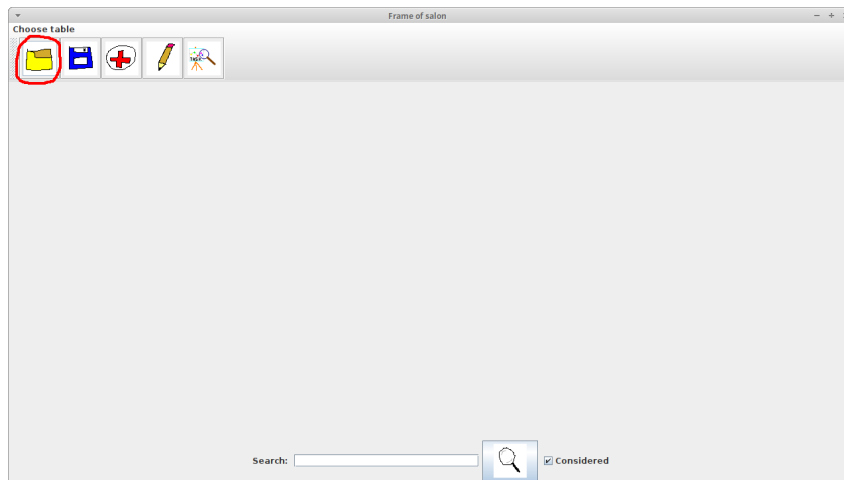


Рисунок 9. Загрузка данных.

Чтобы редактировать или добавить запись, нужно выбрать соответствующую кнопку. Эти кнопки представлены на рисунке 10. Далее заполнить необходимые поля. Пример заполненных полей для типа сервиса представлен на рисунке 11.

Choose table Добавить Редактировать

ID	Service type	Date	Employee	Employee's cut	Client	Price	Relevance
634	Test2 - 23	17.12.2005	Haynes Haywood - 4	0.99	Mcknight Bradley - 48		1 true
240	Massage - 4	01.04.1997	Fitzgerald Jama - 15	658	Ewing Carlita - 2		924 true
581	Nose cleaning - 18	23.12.2005	Benson Stanford - 18	136.817	Sloan Cordell - 11		150 true
547	Manicure - 10	16.12.2005	Cervantes Joel - 7	1.125	Parrish Marielle - 78		1,500 true
348	Ear piercing for childre...	26.11.2009	Benton Lenny - 19	2.759	Mendoza Margaret - 15		3,190 true
674	Pedicures - 6	13.12.2005	Fitzgerald Jama - 15	1,101.57	Wagner Sommer - 65		1,509 true
762	Lawyer defense - 9	18.12.2005	Holden Geoffrey - 20	9,900	Swanson Margorie - 1		10,000 true
544	Nail painting - 3	14.12.2005	Cervantes Joel - 7	390.5	Owens Greg - 143		550 true
661	Contouring plastic - 8	27.12.2005	Lion Goose - 24	1,850	Ware Lecia - 103		2,500 true
425	Smear in the mud - 17	22.02.1955	Rosales Virgilio - 13	2,658	Nichols Timika - 41		2,832 true
112	Manicure - 10	12.01.1986	McLeod Antoine2 - 8	461	Long Babette - 150		891 true
510	Wash hair - 12	21.12.2005	Dyer Annalee - 1	95	Aguilar Francesco - 28		100 true
432	Massage - 4	19.12.1992	McLeod Antoine2 - 8	1,669	Jenkins Lasandra - 64		2,327 true
337	Peeling - 11	20.05.1959	Briggs Laurence - 9	2,301	Bowen Azucena - 132		3,635 true
622	Highlighting - 7	16.12.2005	Alvarez Junko - 17	750	Dwa Aka - 151		1,000 true
427	Lawyer defense - 9	28.06.1954	Holden Geoffrey - 20	1,843	Herring Chia - 120		3,394 true
524	MeowTherapy - 2	20.12.2005	Vazquez Marquis - 16	0.5	Barnett Frederick - 22		50 true
48	Peeling - 11	05.02.2001	Haynes Haywood - 4	3,750	Fuentes Taneka - 121		4,560 true
523	Pedicures - 6	26.12.2005	Benton Lenny - 19	1,101.57	Flores Keith - 126		1,509 true
357	Paint nails - 20	01.10.1956	Cervantes Joel - 7	1,224	Garrett Wilton - 125		1,401 true
588	Pedicures - 6	19.12.2005	Lion Goose - 24	1,101.57	Caldwell Jerald - 16		1,509 true
234	Braiding pigtails - 13	24.11.1980	Alvarez Junko - 17	255	Hunter Mariano - 72		473 true
375	Manicure - 10	29.04.2008	Potter ZulemaZub - 11	276	Yates Randal - 45		285 true
446	Braiding pigtails - 13	28.10.1998	Hahn Moshe - 5	2,820	Perkins Fermin - 140		3,919 true
511	Massage - 4	23.12.2005	Giles Leonardo - 3	1,600	Sellers Macy - 117		2,000 true
678	Smear in the mud - 17	26.12.2005	Rosales Virgilio - 13	11.7	Hebert Willard - 97		13 true
758	Contouring plastic - 8	15.12.2005	Haynes Haywood - 4	1,850	Oliver Zada - 130		2,500 true
24	Haircut with blue lamin...	21.10.1959	Snyder Fausto - 14	2,787	Sherman Ping - 29		4,302 true
153	Depilation - 16	20.05.1973	Briggs Laurence - 9	2,038	Tyson Inell - 23		3,893 true
101	Haircut with blue lamin...	10.01.1978	Potter ZulemaZub - 11	2,419	Sloan Cordell - 11		2,469 true
223	Sugaring - 1	21.02.1963	Blankenship Brady - 21	1,856	Knowles Walker - 56		3,197 true
15	Braiding pigtails - 13	18.10.2008	Rosales Virgilio - 13	3,449	Ross Kizzie - 70		4,590 true


Search:  ☒ Considered

Рисунок 10. Выбор редактирования/добавления.

Edit specialization

ID is 20

Description:

Price:

Percentage:

State: ☒ relevant

Done!

Рисунок 11. Редактирование типа сервиса.

Чтобы сохранить изменения, выберите соответствующую кнопку. Такая кнопка представлена на рисунке 12.

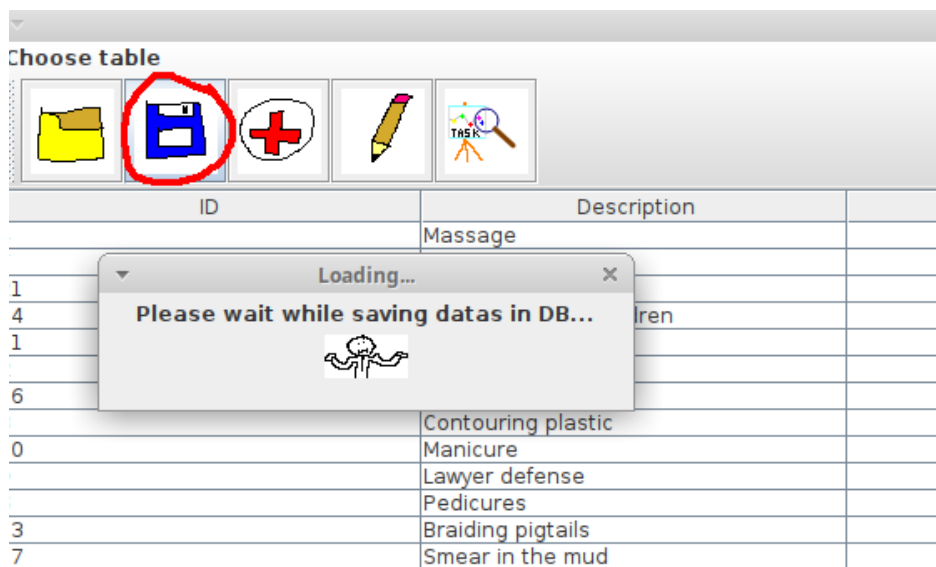


Рисунок 12. Сохранение изменений.

Чтобы просмотреть отчётность о работе салона выберите соответствующую кнопку. В новом окне определите необходимый период времени, после вы сможете выбрать интересующий пункт отчётности. Внизу под таблицей появится статистика работы салона красоты в этот период времени. Чтобы сгенерировать отчёт PDF нажмите соответствующую кнопку сверху справа. См. рисунок 13.

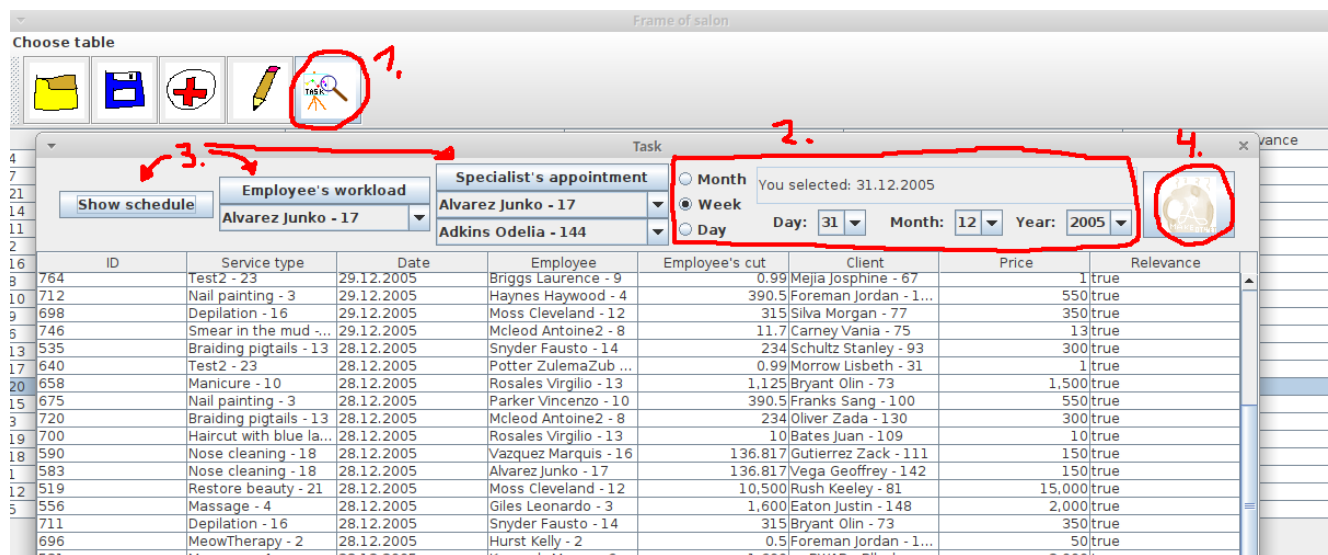


Рисунок 13. Получение отчёта.

Документирование средствами javadoc

С помощью утилиты javadoc была сформирована документация. Примеры такой документации представлены на рисунках 14, 15, 16. Команда для генерации такой документации в интерпретаторе bash: `«javadoc $(find /project/dir -name "*.java")»`.

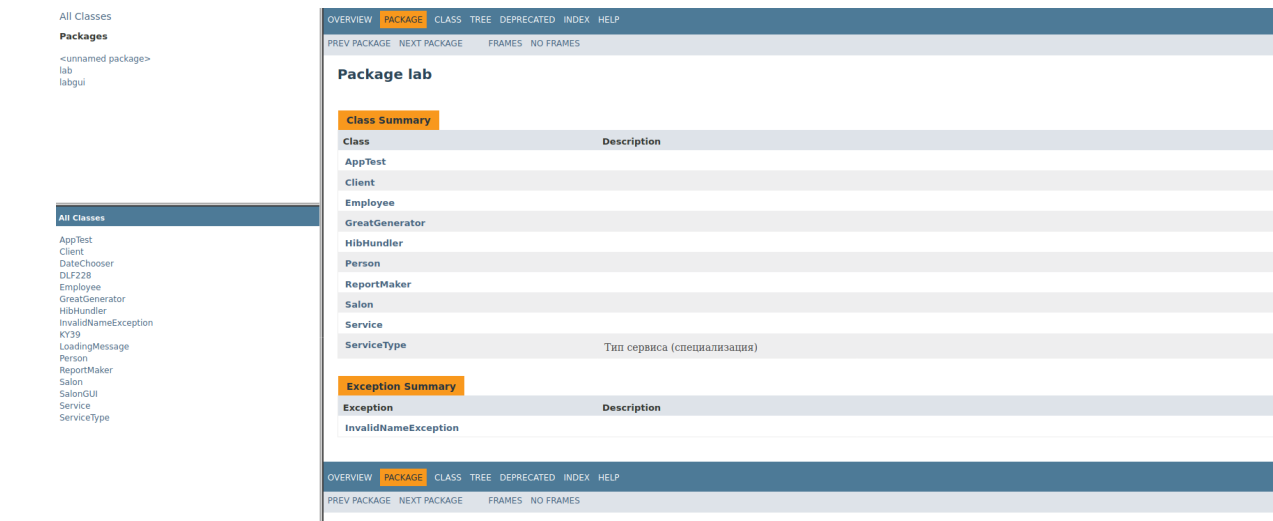


Рисунок 14. Пример документации javadoc 1.

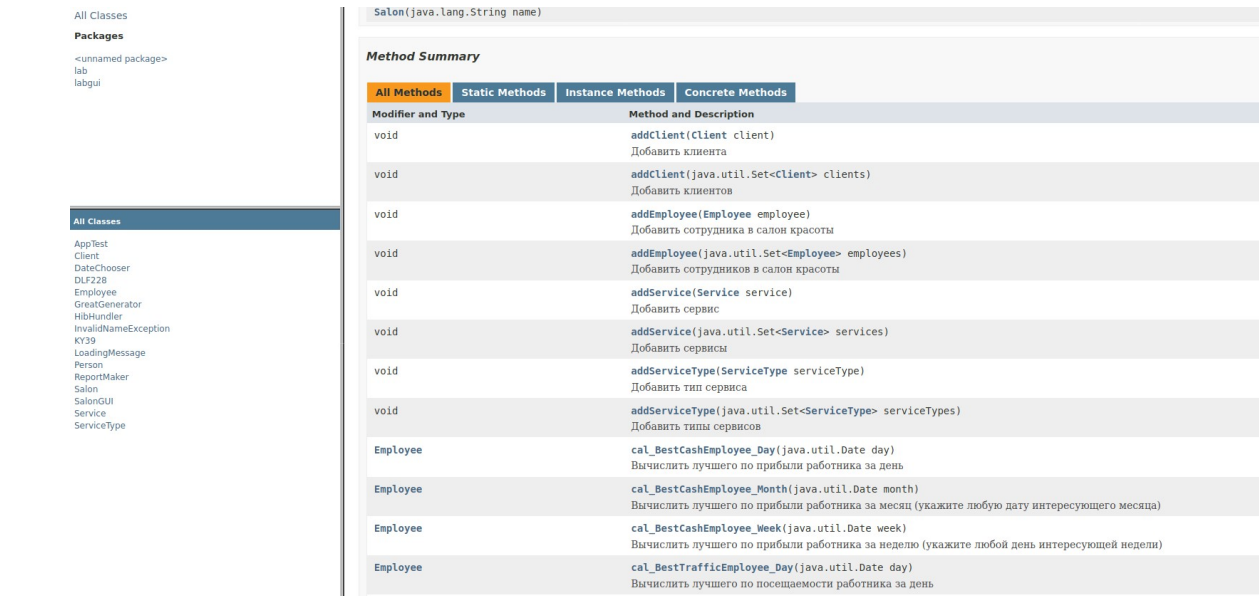


Рисунок 15. Пример документации javadoc 2.

All Classes

Packages

<unnamed package>
lab
labgui

All Classes

AppTest
Client
DateChooser
Employee
GreatGenerator
HibHundler
InvalidNameException
K739
LoadingMessage
Person
ReportMaker
Salon
SalonGUI
Service
ServiceType

Constructors

Constructor and Description

Service()
Service(ServiceType serviceType, java.util.Date dateBegin, Employee whoBo, Client client, Salon forWhichSalon)
Service(ServiceType serviceType, java.util.Date dateBegin, Employee whoBo, double cashReward, Client client, double thenPrice, Salon forWhichSalon)

Method Summary

All Methods

Instance Methods

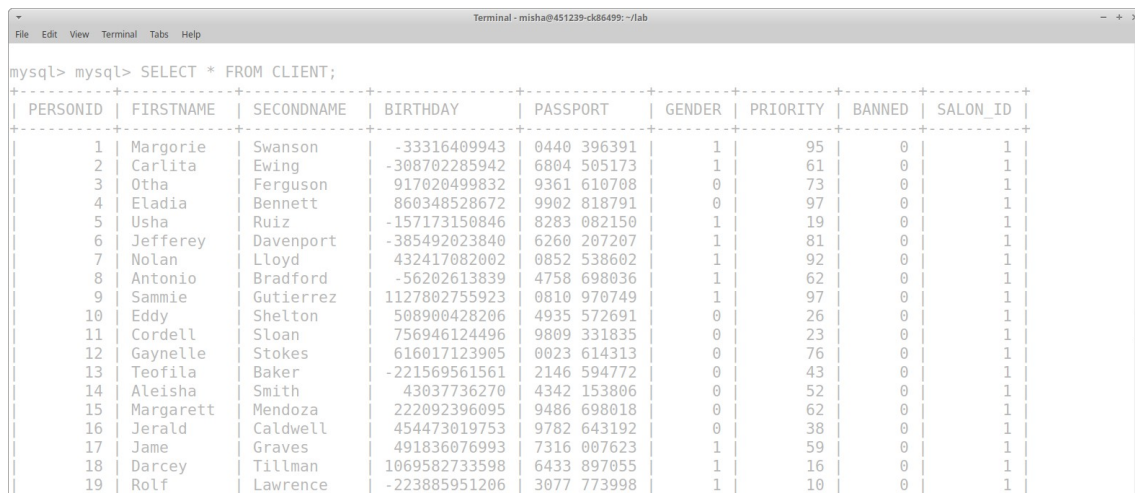
Concrete Methods

Modifier and Type	Method and Description
boolean	equals(java.lang.Object otherObj)
double	getCashReward() Возвращает сумму, которую получил сотрудник после оказания услуги
Client	getClient()
java.util.Date	getDateBegin() Получить дату начала сервиса
Employee	getEmployee() Возвращает работника, который оказывал услугу
int	getId() Получить ID сервиса
double	getPrice() Получить сумму, которую заплатил клиент
ServiceType	getServiceType() Получить тип сервиса
boolean	isRELEVANT() Возможно эта запись создана с ошибкой или ещё какая-нибудь причина, по которой она не должна считаться Если true, то будет учитываться и показываться, если false, то эта запись не в счёт
void	setRELEVANT(boolean rel) Возможно эта запись создана с ошибкой или ещё какая-нибудь причина, по которой она не должна считаться Установите false, чтобы эта запись была не в счёт.
java.lang.String	toString()

Рисунок 16. Пример документации javadoc 3.

Описание базы данных

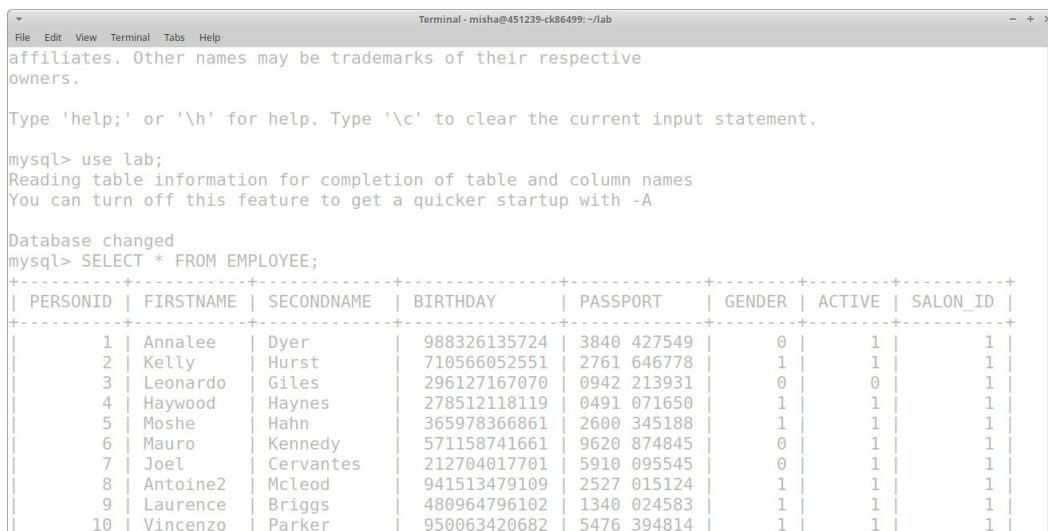
В курсовой работе была использована библиотека для Java hibernate для обращения к СУБД MySQL. Была создана база данных lab и таблицы CLIENT (для класса Client), EMPLOYEE (для класса Employee), SERVICETYPE (для класса ServiceType), SERVICE (для класса Service), SALON (для класса Salon) и таблица EMPLOYEE_SERVICETYPE для организации связи ManyToMany между классами Employee и ServiceType. Таблицы изображены на рисунках 17, 18, 19, 20, 21, 22 соответственно.



```
mysql> SELECT * FROM CLIENT;
```

PERSONID	FIRSTNAME	SECONDDNAME	BIRTHDAY	PASSPORT	GENDER	PRIORITY	BANNED	SALON_ID
1	Margorie	Swanson	-33316409943	0440 396391	1	95	0	1
2	Carlita	Ewing	-308702285942	6804 505173	1	61	0	1
3	Otha	Ferguson	917020499832	9361 610708	0	73	0	1
4	Eladia	Bennett	860348528672	9902 818791	0	97	0	1
5	Usha	Ruiz	-157173150846	8283 082150	1	19	0	1
6	Jefferey	Davenport	-385492023840	6260 207207	1	81	0	1
7	Nolan	Lloyd	432417082002	0852 538602	1	92	0	1
8	Antonio	Bradford	-56202613839	4758 698036	1	62	0	1
9	Sammie	Gutierrez	1127802755923	0810 970749	1	97	0	1
10	Eddy	Shelton	508900428206	4935 572691	0	26	0	1
11	Cordell	Sloan	756946124496	9809 331835	0	23	0	1
12	Gaynelle	Stokes	616017123905	0023 614313	0	76	0	1
13	Teofila	Baker	-221569561561	2146 594772	0	43	0	1
14	Aleisha	Smith	43037736270	4342 153806	0	52	0	1
15	Margarett	Mendoza	222092396095	9486 698018	0	62	0	1
16	Jerald	Caldwell	454473019753	9782 643192	0	38	0	1
17	Jame	Graves	491836076993	7316 007623	1	59	0	1
18	Darcey	Tillman	1069582733598	6433 897055	1	16	0	1
19	Rolf	Lawrence	-223885951206	3077 773998	1	10	0	1

Рисунок 17. Таблица MySQL CLIENT.



```
mysql> use lab;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> SELECT * FROM EMPLOYEE;
```

PERSONID	FIRSTNAME	SECONDDNAME	BIRTHDAY	PASSPORT	GENDER	ACTIVE	SALON_ID
1	Annalee	Dyer	988326135724	3840 427549	0	1	1
2	Kelly	Hurst	710566052551	2761 646778	1	1	1
3	Leonardo	Giles	296127167070	0942 213931	0	0	1
4	Haywood	Haynes	278512118119	0491 071650	1	1	1
5	Moshe	Hahn	365978366861	2600 345188	1	1	1
6	Mauro	Kennedy	571158741661	9620 874845	0	1	1
7	Joel	Cervantes	212704017701	5910 095545	0	1	1
8	Antoine2	McLeod	941513479109	2527 015124	1	1	1
9	Laurence	Briggs	480964796102	1340 024583	1	1	1
10	Vincenzo	Parker	950063420682	5476 394814	1	1	1

Рисунок 18. Таблица MySQL EMPLOYEE.

Terminal - misha@451239-ck86499: ~/lab

808 rows in set (0.00 sec)

```
mysql> SELECT * FROM SERVICETYPE;
```

SERVICETYPEID	DESCRIPTION	CURRENTPRICE	PERCENTAGETOEMPLOYEE	RELEVANT	SALON_ID
1	Sugaring	250.0000	85.0000	1	1
2	MeowTherapy	50.0000	1.0000	1	1
3	Nail painting	550.0000	71.0000	1	1
4	Massage	2000.0000	80.0000	1	1
5	Makeup	500.0000	70.0000	1	1
6	Pedicures	1509.0000	73.0000	1	1
7	Highlighting	1000.0000	75.0000	1	1
8	Contouring plastic	2500.0000	74.0000	1	1
9	Lawyer defense	10000.0000	99.0000	1	1
10	Manicure	1500.0000	75.0000	1	1
11	Peeling	450.0000	70.0000	1	1
12	Wash hair	100.0000	95.0000	1	1
13	Braiding pigtails	300.0000	78.0000	1	1
14	Ear piercing for children	700.0000	75.0000	1	1
15	Haircut with blue laminate	10.0000	100.0000	1	1
16	Depilation	350.0000	90.0000	1	1
17	Smear in the mud	13.0000	90.0000	1	1
18	Nose cleaning	150.0000	91.2112	1	1
19	Mesotherapy	3000.0000	69.0000	1	1
20	Paint nails	500.0000	85.0000	1	1
21	Restore beauty	15000.0000	70.0000	1	1

Рисунок 19. Таблица MySQL SERVICETYPE.

Terminal - misha@451239-ck86499: ~/lab

155 rows in set (0.01 sec)

```
mysql> mysql> SELECT * FROM SERVICE;
```

SERVICEID	DATEBEGINI	CASHREWARD	THENPRICE	REL	SERVICETYPE_ID	CLIENT_ID	EMPLOYEE_ID	SALON_ID
1	792366741683	1717.0000	2240.0000	1	4	129	18	1
2	-113663035434	930.0000	1392.0000	1	8	83	9	1
3	153724786302	591.0000	786.0000	1	4	87	15	1
4	495771433100	2415.0000	4738.0000	1	9	38	5	1
5	-278676578235	2466.0000	3037.0000	1	10	76	21	1
6	841286311279	956.0000	1203.0000	1	11	100	21	1
7	-480627484915	548.0000	557.0000	1	2	138	14	1
8	-20968664751	247.0000	251.0000	1	1	72	4	1
9	-381766452547	1690.0000	1744.0000	1	11	90	8	1
10	547185613052	2942.0000	3322.0000	1	12	9	4	1
11	-447384610415	3707.0000	4986.0000	1	10	65	4	1
12	728159791923	2352.0000	2878.0000	1	13	79	8	1
13	756596486708	2522.0000	3086.0000	1	14	116	15	1
14	715332061602	424.0000	566.0000	1	6	73	11	1
15	1224288775012	3449.0000	4590.0000	1	13	70	13	1
16	245280290763	1064.0000	1195.0000	1	15	134	9	1
17	-212377801874	2472.0000	3730.0000	1	14	118	8	1
18	450867000375	3220.0000	4978.0000	1	16	97	16	1
19	670522752105	1154.0000	2240.0000	1	4	27	11	1

Рисунок 20. Таблица MySQL SERVICE.

```
Terminal - misha@451239-ck86499: ~/lab
File Edit View Terminal Tabs Help

+-----+
| 4 | 19 |
| 20 | 19 |
| 20 | 19 |
| 22 | 2 |
| 22 | 1 |
| 22 | 3 |
| 22 | 2 |
| 22 | 1 |
| 22 | 3 |
| 23 | 2 |
| 23 | 1 |
| 23 | 3 |
| 23 | 2 |
| 23 | 1 |
| 23 | 3 |
+-----+
164 rows in set (0.01 sec)

mysql> SELECT * FROM SALON;
+-----+
| SALONID | SALONNAME |
+-----+
| 1 | Salon Harry Dubua face |
+-----+
1 row in set (0.00 sec)

mysql>
```

Рисунок 21. Таблица MySQL SALON.

```
Terminal - misha@451239-ck86499: ~/lab
File Edit View Terminal Tabs Help

mysql> mysql> SELECT * FROM EMPLOYEE_SERVICETYPE;
+-----+
| M2M_EMPLOYEE_ID | M2M_SERVICETYPE_ID |
+-----+
| 1 | 1 |
| 1 | 6 |
| 1 | 1 |
| 2 | 1 |
| 6 | 1 |
| 11 | 1 |
| 9 | 1 |
| 17 | 1 |
| 19 | 1 |
| 15 | 1 |
| 16 | 1 |
| 2 | 2 |
| 2 | 1 |
| 2 | 3 |
| 2 | 6 |
| 2 | 4 |
| 2 | 5 |
| 2 | 2 |
| 3 | 2 |
| 4 | 2 |
| 11 | 2 |
| 12 | 2 |
| 19 | 2 |
+-----+
```

Рисунок 22. Таблица MySQL EMPLOYEE_SERVICETYPE.

Листинг команд для создания БД в СУБД MySQL:

```
CREATE DATABASE lab;
```

```
use lab;
```

```
CREATE TABLE `CLIENT` (  
  `PERSONID` int(11) NOT NULL AUTO_INCREMENT,  
  `FIRSTNAME` varchar(50) NOT NULL,  
  `SECONDNAME` varchar(50) NOT NULL,  
  `BIRTHDAY` BIGINT NOT NULL,  
  `PASSPORT` varchar(50) NOT NULL,  
  `GENDER` int(11) NOT NULL,  
  `PRIORITY` int(11) NOT NULL,  
  `BANNED` int(11) NOT NULL,  
  `SALON_ID` int(11),  
  PRIMARY KEY (`PERSONID`)  
) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=latin1;
```

```
CREATE TABLE `EMPLOYEE` (  
  `PERSONID` int(11) NOT NULL AUTO_INCREMENT,  
  `FIRSTNAME` varchar(50) NOT NULL,  
  `SECONDNAME` varchar(50) NOT NULL,  
  `BIRTHDAY` BIGINT NOT NULL,  
  `PASSPORT` varchar(50) NOT NULL,  
  `GENDER` int(11) NOT NULL,  
  `ACTIVE` int(11) NOT NULL,  
  `SALON_ID` int(11),  
  PRIMARY KEY (`PERSONID`)  
) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=latin1;
```

```
CREATE TABLE `SERVICETYPE` (  
  `SERVICETYPEID` int(11) NOT NULL AUTO_INCREMENT,  
  `DESCRIPTION` varchar(50) NOT NULL,  
  `CURRENTPRICE` double(11, 4) NOT NULL,  
  `PERCENTAGETOEMPLOYEE` double(11, 4) NOT NULL,  
  `RELEVANT` int(11) NOT NULL,  
  `SALON_ID` int(11),  
  PRIMARY KEY (`SERVICETYPEID`)  
) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=latin1;
```

```
CREATE TABLE `SERVICE` (  
  `SERVICEID` int(11) NOT NULL AUTO_INCREMENT,  
  `DATEBEGIN` BIGINT NOT NULL,  
  `CASHREWARD` double(11, 4) NOT NULL,  
  `THENPRICE` double(11, 4) NOT NULL,  
  `REL` int(11) NOT NULL,  
  `SERVICETYPE_ID` int(11) NOT NULL,
```

```
`CLIENT_ID` int(11) NOT NULL,  
`EMPLOYEE_ID` int(11) NOT NULL,  
`SALON_ID` int(11),  
PRIMARY KEY (`SERVICEID`)  
) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=latin1;
```

```
CREATE TABLE `SALON` (  
  `SALONID` int(11) NOT NULL AUTO_INCREMENT,  
  `SALONNAME` varchar(50) NOT NULL,  
  PRIMARY KEY (`SALONID`)  
) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=latin1;
```

```
CREATE TABLE `EMPLOYEE_SERVICETYPE` (  
  `M2M_EMPLOYEE_ID` int(11) NOT NULL,  
  `M2M_SERVICETYPE_ID` int(11) NOT NULL,  
  CONSTRAINT FK_M2M_EMPLOYEE_ID FOREIGN KEY (M2M_EMPLOYEE_ID)  
    REFERENCES EMPLOYEE (PERSONID),  
  CONSTRAINT FK_M2M_SERVICETYPE_ID FOREIGN KEY (M2M_SERVICETYPE_ID)  
    REFERENCES SERVICETYPE (SERVICETYPEID)  
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

```
CREATE USER 'LABUSER' IDENTIFIED BY '[пароль]';  
GRANT ALL PRIVILEGES ON lab.* TO 'LABUSER';  
FLUSH PRIVILEGES;
```

Заключение

В результате проделанной работы разработан ПК, предназначенный для администрирования работы салона красоты. В процессе проектирования ПК созданы прототип пользовательского графического интерфейса, диаграмма классов, руководство пользователю. Было закреплено теоретические знания, приобретены практические навыки по проектированию и разработке программного обеспечения на объектно-ориентированном языке Java.

ПРИЛОЖЕНИЕ А.

Исходный текст программы:

Salon.java

```
package lab;

import lab.*;

import java.lang.*;
import java.util.*;

import javax.persistence.*;

/**
 * Салон красоты
 */
@Entity
@Table(name="SALON")
public class Salon
{
    @Id
    @GeneratedValue(strategy=GenerationType.IDENTITY)
    @Column(name="SALONID")
    private int salonID;

    @Column(name="SALONNAME")
    private String salonName;

    @OneToMany(fetch = FetchType.LAZY, mappedBy = "salon", cascade = CascadeType.ALL)
    private Set<ServiceType> serviceTypes;

    @OneToMany(fetch = FetchType.LAZY, mappedBy = "salon", cascade = CascadeType.ALL)
    private Set<Service> allServices;

    @OneToMany(fetch = FetchType.LAZY, mappedBy = "salon", cascade = CascadeType.ALL)
    private Set<Employee> allEmployees;

    @OneToMany(fetch = FetchType.LAZY, mappedBy = "salon", cascade = CascadeType.ALL)
    private Set<Client> allClients;

    public Salon(){init4Const();}

    public Salon(String name)
```

```
{
init4Const();
this.salonName = name;
}
```

```
private void init4Const()
{
this.serviceTypes = new HashSet<ServiceType>();
this.allServices = new HashSet<Service>();
this.allEmployees = new HashSet<Employee>();
this.allClients = new HashSet<Client>();
}
```

```
//Сотрудники
```

```
/**
```

```
* Добавить сотрудника в салон красоты
```

```
*
```

```
* @param employee сотрудник, принимаемый на работу
```

```
*/
```

```
public synchronized void addEmployee(Employee employee)
```

```
{
```

```
if(!allEmployees.contains(employee))
```

```
this.allEmployees.add(employee);
```

```
}
```

```
/**
```

```
* Добавить сотрудников в салон красоты
```

```
*
```

```
* @param employees сотрудники, принимаемые на работу
```

```
*/
```

```
public synchronized void addEmployee(Set<Employee> employees)
```

```
{
```

```
for(Employee e : employees)
```

```
if(!this.allEmployees.contains(e))
```

```
this.allEmployees.add(e);
```

```
}
```

```
/**
```

```
* Удалить сотрудника из салона красоты
```

```
*
```

```
* @param employee удаляемый сотрудник
```

```
*/
```

```
public synchronized void fireEmployee(Employee employee)
```

```
{
```

```
this.allEmployees.remove(employee);
```

```
}
```

//Клиенты

/**

** Добавить клиента*

** @param client добавляемый клиент*

**/*

public synchronized void addClient(**Client** client)

{

if(!this.allClients.contains(client))

this.allClients.add(client);

}

/**

** Добавить клиентов*

** @param clients добавляемые клиенты*

**/*

public synchronized void addClient(**Set**<**Client**> clients)

{

for(**Client** c : clients)

if(!this.allClients.contains(c))

this.allClients.add(c);

}

//Сервисы

/**

** Добавить сервис*

** @param service добавляемый сервис*

**/*

public synchronized void addService(**Service** service)

{

if(!this.allServices.contains(service))

this.allServices.add(service);

}

/**

** Добавить сервисы*

** @param services добавляемые сервисы*

**/*

public synchronized void addService(**Set**<**Service**> services)

{

for(**Service** s : services)

if(!this.allServices.contains(s))

this.allServices.add(s);

}


```

//Типы сервисов
/**
 * Добавить тип сервиса
 *
 * @param serviceType добавляемый тип сервиса
 */
public synchronized void addServiceType(ServiceType serviceType)
{
    if(!this.serviceTypes.contains(serviceType))
    this.serviceTypes.add(serviceType);
}

/**
 * Добавить типы сервисов
 *
 * @param serviceTypes добавляемый типы сервисов
 */
public synchronized void addServiceType(Set<ServiceType> serviceTypes)
{
    for(ServiceType st : serviceTypes)
    if(!this.serviceTypes.contains(st))
    this.serviceTypes.add(st);
}

/**
 * Получить название салона красоты
 *
 * @return название салона красоты
 */
public String getName()
{
    return this.salonName;
}

/**
 * Получить сотрудников
 *
 * @return сотрудники салона
 */
public synchronized Set<Employee> getEmployees()
{
    return new HashSet<Employee>(this.allEmployees);
}

/**
 * Получить клиентов

```

```

*
* @return клиенты салона
*/
public synchronized Set<Client> getClients()
{
return new HashSet<Client>(this.allClients);
}

/**
* Получить сервисы
*
* @return проведённые сервисы
*/
public synchronized Set<Service> getDeals()
{
return new HashSet<Service>(this.allServices);
}

/**
* Получить типы сервисов
*
* @return проводимые типы сервисов
*/
public synchronized Set<ServiceType> getProvidedServices()
{
return new HashSet<ServiceType>(this.serviceTypes);
}

/**
* Получить ID салона красоты
*
* @return ID салона красоты
*/
public int getID()
{
return this.salonID;
}

/**
* Инициализация салона
*
* @return количество элементов в салоне
*/
public synchronized int initAllLazyRecords()
{
int counter = 0;
for(Employee e : allEmployees)

```

```

{
++counter;
e.getMasteredServices();
}
for(Client c : allClients)
++counter;
for(Service s : allServices)
++counter;
for(ServiceType st : serviceTypes)
{
++counter;
st.getWhoMasteredThisServiceTypes();
}
return counter;
}

```

```
//
```

```

=====Задания=====
=====
=====

```

```
/**
```

```

* Посмотреть все услуги, которые проводились в день day (укажите любое время этого
дня)

```

```
*/
```

```
@SuppressWarnings( "deprecation" )
```

```
public synchronized Set<Service> showServicesPerDay(Date day)
```

```
{
```

```
int neededDay = day.getDate();
```

```
Set<Service> res = new
```

```
TreeSet<Service>(GreatGenerator.getServicesDateComparator());
```

```
for(Service s : allServices)
```

```
if(s.getDateBegin().getMonth() == day.getMonth() && s.getDateBegin().getYear() ==
day.getYear() && s.isRELEVANT() == true)
```

```
if(s.getDateBegin().getDate() == neededDay)
```

```
res.add(s);
```

```
return res;
```

```
}
```

```
/**
```

```

* Посмотреть все услуги, которые проводились в неделю week (укажите любое время
этой недели)

```

```
*/
```

```
@SuppressWarnings( "deprecation" )
```

```
public synchronized Set<Service> showServicesPerWeek(Date week)
```

```
{
```

```

Set<Service> res = new
TreeSet<Service>(GreatGenerator.getServicesDateComporator());
Date[] ds = getWeekKnowsDay(week);
for(Service s : allServices)
if(s.getDateBegin().getMonth() == week.getMonth() && s.getDateBegin().getYear() ==
week.getYear() && s.isRELEVANT() == true)
if(
s.getDateBegin().getDate() == ds[0].getDate()
|| s.getDateBegin().getDate() == ds[1].getDate()
|| s.getDateBegin().getDate() == ds[2].getDate()
|| s.getDateBegin().getDate() == ds[3].getDate()
|| s.getDateBegin().getDate() == ds[4].getDate()
|| s.getDateBegin().getDate() == ds[5].getDate()
|| s.getDateBegin().getDate() == ds[6].getDate()
)
res.add(s);
return res;
}

/**
 * Получить дни недели
 *
 * @param oneDayOfThisWeek день недели
 * @return массив дней, принадлежащей недели
 */
@SuppressWarnings( "deprecation" )
public synchronized static Date[] getWeekKnowsDay(Date oneDayOfThisWeek)
{
Date one = new Date(oneDayOfThisWeek.getTime());
if(one.getDay() == 0)
one.setDate(one.getDate()-6);
else
{
one.setDate(one.getDate()-(one.getDay()-1));
}
Date[] ds = new Date[7];
Date buff;
for(int i = 0; i < 7; ++i)
{
buff = new Date(one.getTime());
buff.setDate(buff.getDate()+i);
ds[i] = buff;
}
return ds;
}

/**

```

```
* Посмотреть все услуги, которые проводились в месяц month (укажите любое время этого месяца)
```

```
*/
```

```
@SuppressWarnings( "deprecation" )
```

```
public synchronized Set<Service> showServicesPerMonth(Date month)
```

```
{
```

```
    int neededMonth = month.getMonth();
```

```
    Set<Service> res = new
```

```
    TreeSet<Service>(GreatGenerator.getServicesDateComparator());
```

```
    for(Service s : allServices)
```

```
        if(s.getDateBegin().getYear() == month.getYear() && s.isRELEVANT() == true)
```

```
            if(s.getDateBegin().getMonth() == neededMonth)
```

```
                res.add(s);
```

```
    return res;
```

```
}
```

```
/**
```

```
* Вычислить загруженность сотрудника за месяц month (укажите любую дату из этого месяца)
```

```
*/
```

```
public Set<Service> calculateWorkloadMonth(Date month, Employee employee)
```

```
{
```

```
    Set<Service> res = new
```

```
    TreeSet<Service>(GreatGenerator.getServicesDateComparator());
```

```
    Set<Service> perMonth = showServicesPerMonth(month);
```

```
    for(Service s : perMonth)
```

```
        if(s.getEmployee().getID() == employee.getID())
```

```
            res.add(s);
```

```
    return res;
```

```
}
```

```
/**
```

```
* Вычислить загруженность сотрудника за неделю week (укажите любой день интересующей недели)
```

```
*/
```

```
public Set<Service> calculateWorkloadWeek(Date week, Employee employee)
```

```
{
```

```
    Set<Service> res = new
```

```
    TreeSet<Service>(GreatGenerator.getServicesDateComparator());
```

```
    Set<Service> perWeek = showServicesPerWeek(week);
```

```
    for(Service s : perWeek)
```

```
        if(s.getEmployee().getID() == employee.getID())
```

```
            res.add(s);
```

```
    return res;
```

```
}
```

```

/**
 * Вычислить загруженность сотрудника за день day (укажите любое время интересую-
 щего дня)
 */
public Set<Service> calculateWorkloadDay(Date day, Employee employee)
{
    Set<Service> res = new
    TreeSet<Service>(GreatGenerator.getServicesDateComporator());
    Set<Service> perDay = showServicesPerDay(day);
    for(Service s : perDay)
        if(s.getEmployee().getID() == employee.getID())
            res.add(s);
    return res;
}

```

```

/**
 * Вычислить клиента client специалисту employee за месяц month (укажите любую дату
 из этого месяца)
 */
public Set<Service> calculateClientRaspMonth(Date month, Employee employee, Client
client)
{
    Set<Service> res = new
    TreeSet<Service>(GreatGenerator.getServicesDateComporator());
    Set<Service> perMonth = showServicesPerMonth(month);
    for(Service s : perMonth)
        if(s.getEmployee().getID() == employee.getID() && s.getClient().getID() == client.getID())
            res.add(s);
    return res;
}

```

```

/**
 * Вычислить клиента client специалисту employee за неделю week (укажите любой день
 интересующей недели)
 */
public Set<Service> calculateClientRaspWeek(Date week, Employee employee, Client
client)
{
    Set<Service> res = new
    TreeSet<Service>(GreatGenerator.getServicesDateComporator());
    Set<Service> perWeek = showServicesPerWeek(week);
    for(Service s : perWeek)
        if(s.getEmployee().getID() == employee.getID() && s.getClient().getID() == client.getID())
            res.add(s);
    return res;
}

```

```

/**
 * Вычислить клиента client специалисту employee за день day (укажите любое время
 интересующего дня)
 */
public Set<Service> calculateClientRaspDay(Date day, Employee employee, Client
client)
{
    Set<Service> res = new
    TreeSet<Service>(GreatGenerator.getServicesDateComporator());
    Set<Service> perDay = showServicesPerDay(day);
    for(Service s : perDay)
    if(s.getEmployee().getID() == employee.getID() && s.getClient().getID() == client.getID())
    res.add(s);
    return res;
}

```

```

/**
 * Посчитать клиентов за месяц (укажите любую дату интересующего месяца)
 * Если unique == true, то покажет только уникальных клиентов
 */
public int calculateClientsNumMonth(Date month, boolean unique)
{
    int resInt;
    Set<Client> res = new HashSet<Client>();
    Set<Service> perMonth = showServicesPerMonth(month);
    resInt = 0;
    for(Service s : perMonth)
    {
        res.add(s.getClient());
        ++resInt;
    }
    if(unique == true)
    return res.size();
    else
    return resInt;
}

```

```

/**
 * Посчитать клиентов за неделю (укажите любой день интересующей недели)
 * Если unique == true, то покажет только уникальных клиентов
 */
public int calculateClientsNumWeek(Date week, boolean unique)
{
    int resInt;
    Set<Client> res = new HashSet<Client>();
    Set<Service> perWeek = showServicesPerWeek(week);
    resInt = 0;

```

```

for(Service s : perWeek)
{
res.add(s.getClient());
++resInt;
}
if(unique == true)
return res.size();
else
return resInt;
}

```

```

/**
 * Посчитать клиентов за день
 * Если unique == true, то покажет только уникальных клиентов
 */

```

```

public int calculateClientsNumDay(Date day, boolean unique)
{
int resInt;
Set<Client> res = new HashSet<Client>();
Set<Service> perDay = showServicesPerDay(day);
resInt = 0;
for(Service s : perDay)
{
res.add(s.getClient());
++resInt;
}
if(unique == true)
return res.size();
else
return resInt;
}

```

```

/**
 * Вычислить приход денег за месяц (укажите любую дату интересующего месяца)
 * Если percent == true, то вычислит доход с учётом выплаченных процентов сотрудни-
кам, иначе полную сумму
 */

```

```

public double calculateIncomeCashMonth(Date month, boolean percent)
{
double res;
Set<Service> perMonth = showServicesPerMonth(month);
res = 0;
for(Service s : perMonth)
res += percent==true?(s.getPrice()-s.getCashReward()):s.getPrice();
return res;
}

```



```
/**
 * Вычислить приход денег за неделю (укажите любой день интересующей недели)
 * Если percent == true, то вычислит доход с учётом выплаченных процентов сотрудни-
 кам, иначе полную сумму
 */
```

```
public double calculateIncomeCashWeek(Date week, boolean percent)
{
double res;
Set<Service> perWeek = showServicesPerWeek(week);
res = 0;
for(Service s : perWeek)
res += percent==true?(s.getPrice()-s.getCashReward()):s.getPrice();
return res;
}
```

```
/**
 * Вычислить приход денег за день
 * Если percent == true, то вычислит доход с учётом выплаченных процентов сотрудни-
 кам, иначе полную сумму
 */
```

```
public double calculateIncomeCashDay(Date day, boolean percent)
{
double res;
Set<Service> perDay = showServicesPerDay(day);
res = 0;
for(Service s : perDay)
res += percent==true?(s.getPrice()-s.getCashReward()):s.getPrice();
return res;
}
```

```
/**
 * Вычислить лучшего по прибыли работника за месяц (укажите любую дату интересу-
 щего месяца)
 */
```

```
public Employee cal_BestCashEmployee_Month(Date month)
{
HashMap<Employee, Double> dict = new HashMap<Employee, Double>();
Set<Service> perMonth = showServicesPerMonth(month);
for(Service s : perMonth)
dict.put(s.getEmployee(), Double.valueOf( 0 ));
for(Service s : perMonth)
dict.put(s.getEmployee(), Double.valueOf( dict.get(s.getEmployee()).doubleValue() +
(s.getPrice() - s.getCashReward()) ));
Employee res = null;
double max = -1;
Set<Employee> es = dict.keySet();
for(Employee e : es)
```

```

if(max < dict.get(e).doubleValue())
{
max = dict.get(e).doubleValue();
res = e;
}
return res;
}

```

```

/**

```

```

 * Вычислить лучшего по прибыли работника за неделю (укажите любой день интересу-
 * ющей недели)
 */

```

```

public Employee cal_BestCashEmployee_Week(Date week)
{
HashMap<Employee, Double> dict = new HashMap<Employee, Double>();
Set<Service> perWeek = showServicesPerWeek(week);
for(Service s : perWeek)
dict.put(s.getEmployee(), Double.valueOf( 0 ));
for(Service s : perWeek)
dict.put(s.getEmployee(), Double.valueOf( dict.get(s.getEmployee()).doubleValue() +
(s.getPrice() - s.getCashReward()) ));
Employee res = null;
double max = -1;
Set<Employee> es = dict.keySet();
for(Employee e : es)
if(max < dict.get(e).doubleValue())
{
max = dict.get(e).doubleValue();
res = e;
}
return res;
}

```

```

/**

```

```

 * Вычислить лучшего по прибыли работника за день
 */

```

```

public Employee cal_BestCashEmployee_Day(Date day)
{
HashMap<Employee, Double> dict = new HashMap<Employee, Double>();
Set<Service> perDay = showServicesPerDay(day);
for(Service s : perDay)
dict.put(s.getEmployee(), Double.valueOf( 0 ));
for(Service s : perDay)
dict.put(s.getEmployee(), Double.valueOf( dict.get(s.getEmployee()).doubleValue() +
(s.getPrice() - s.getCashReward()) ));
Employee res = null;
double max = -1;

```

```

Set<Employee> es = dict.keySet();
for(Employee e : es)
if(max < dict.get(e).doubleValue())
{
max = dict.get(e).doubleValue();
res = e;
}
return res;
}

```

```
/**
```

```

* Вычислить лучшего по посещаемости работника за месяц (укажите любую дату ин-
тересующего месяца)
*/

```

```
*/
```

```

public Employee cal_BestTrafficEmployee_Month(Date month)
{
HashMap<Employee, Integer> dict = new HashMap<Employee, Integer>();
Set<Service> perMonth = showServicesPerMonth(month);
for(Service s : perMonth)
dict.put(s.getEmployee(), Integer.valueOf( 0 ));
for(Service s : perMonth)
dict.put(s.getEmployee(), Integer.valueOf( dict.get(s.getEmployee()).intValue() + 1 ));
Employee res = null;
double max = -1;
Set<Employee> es = dict.keySet();
for(Employee e : es)
if(max < dict.get(e).intValue())
{
max = dict.get(e).intValue();
res = e;
}
return res;
}

```

```
/**
```

```

* Вычислить лучшего по посещаемости работника за неделю (укажите любой день ин-
тересующей недели)
*/

```

```
*/
```

```

public Employee cal_BestTrafficEmployee_Week(Date week)
{
HashMap<Employee, Integer> dict = new HashMap<Employee, Integer>();
Set<Service> perWeek = showServicesPerWeek(week);
for(Service s : perWeek)
dict.put(s.getEmployee(), Integer.valueOf( 0 ));
for(Service s : perWeek)
dict.put(s.getEmployee(), Integer.valueOf( dict.get(s.getEmployee()).intValue() + 1 ));
Employee res = null;

```

```

double max = -1;
Set<Employee> es = dict.keySet();
for(Employee e : es)
if(max < dict.get(e).intValue())
{
max = dict.get(e).intValue();
res = e;
}
return res;
}

/**
 * Вычислить лучшего по посещаемости работника за день
 */
public Employee cal_BestTrafficEmployee_Day(Date day)
{
HashMap<Employee, Integer> dict = new HashMap<Employee, Integer>();
Set<Service> perDay = showServicesPerDay(day);
for(Service s : perDay)
dict.put(s.getEmployee(), Integer.valueOf( 0 ));
for(Service s : perDay)
dict.put(s.getEmployee(), Integer.valueOf( dict.get(s.getEmployee()).intValue() + 1 ));
Employee res = null;
double max = -1;
Set<Employee> es = dict.keySet();
for(Employee e : es)
if(max < dict.get(e).intValue())
{
max = dict.get(e).intValue();
res = e;
}
return res;
}

public String toString()
{
StringBuilder s;
synchronized(this)
{
s = new StringBuilder();

s.append("Salon " + this.getID() + " " + this.getName() + ": \n");

s.append("\nCan provide services: \n");
for(ServiceType st : serviceTypes)
s.append("\t" + st + "\n");
}
}

```

```
s.append("\nHave employees: \n");
for(Employee e : allEmployees)
s.append("\t" + e + "\n");
```

```
s.append("\nHad clients: \n");
for(Client c : allClients)
s.append("\t" + c + "\n");
```

```
s.append("\nProvided services: \n");
for(Service se : allServices)
s.append("\t" + se + "\n");
```

```
s.append("\n");
}
```

```
return s.toString();
}
```

@Override

```
public boolean equals(Object otherObj)
{
if (otherObj == this) return true;
if (otherObj == null) return false;
if( this.getClass() != otherObj.getClass() ) return false;
Salon other = (Salon)otherObj;
return this == other;
//return this.getID() == other.getID();
}
```

```
/**
```

```
 * Получить сотрудника по его ID
```

```
 *
```

```
 * @param ID ид сотрудника
```

```
 * @return сотрудник
```

```
 */
```

```
public synchronized Employee getEmployeeByID(int ID)
{
for(Employee e : allEmployees)
if(e.getID() == ID)
return e;
return null;
}
```

```
/**
```

```
 * Получить клиента по его ID
```

```
 *
```

```
 * @param ID ид клиента
```

```

* @return клиент
*/
public synchronized Client getClientByID(int ID)
{
for(Client c : allClients)
if(c.getID() == ID)
return c;
return null;
}

/**
* Получить сервис по его ID
*
* @param ID ид сервиса
* @return сервис
*/
public synchronized Service getServiceByID(int ID)
{
for(Service s : allServices)
if(s.getID() == ID)
return s;
return null;
}

/**
* Получить тип сервиса по его ID
*
* @param ID ид типа сервиса
* @return тип сервиса
*/
public synchronized ServiceType getServiceTypeByID(int ID)
{
for(ServiceType st : serviceTypes)
if(st.getID() == ID)
return st;
return null;
}

/**
* Генерирует рандомный новый сервис и добавляет его в салон красоты
*
* @param from начиная с какой даты
* @param to какой датой заканчивая
*/
@SuppressWarnings( "deprecation" )
public void genNewService(Date from, Date to)
{

```

```

Random r = new Random();
long from_i = from.getTime();
long to_i = to.getTime();
long cur_i = genLong(to_i-from_i) + from_i;

int st_id = r.nextInt(serviceTypes.size()+1);
int e_id = r.nextInt(allEmployees.size()+1);
int c_id = r.nextInt(allClients.size()+1);
ServiceType st = this.getServiceTypeByID(st_id);

```

```

Service s = new Service(
st,
new Date(cur_i),
this.getEmployeeByID(e_id),
st.getCurrentPrice() * st.getPercent()*0.01,
this.getClientByID(c_id),
st.getCurrentPrice(),
this
);
synchronized(this)
{
this.addService(s);
}
System.out.println(s);
}

```

```

private long genLong(long n)
{
// error checking and 2^x checking removed for simplicity.
Random rng = new Random();
long bits, val;
do {
bits = (rng.nextLong() << 1) >>> 1;
val = bits % n;
} while (bits-val+(n-1) < 0L);
return val;
}

```

```

/**
 * Преобразует дату в красиво читаемый вид
 *
 * @param d дата для преобразования
 * @return строка, представляющей дату
 */
@SuppressWarnings( "deprecation" )
public static String doNiceDate(Date d)
{

```

```

String toOut = (d.getYear()+1900) + ".";
toOut += (d.getMonth()+1>9?d.getMonth()+1:"0"+(d.getMonth()+1)) + ".";
toOut += (d.getDate()>9?d.getDate():"0"+d.getDate()) + " ";
toOut += (d.getHours()>9?d.getHours():"0"+d.getHours()) + ":";
toOut += (d.getMinutes()>9?d.getMinutes():"0"+d.getMinutes()) + ":";
toOut += (d.getSeconds()>9?d.getSeconds():"0"+d.getSeconds());
return toOut;
}

```

```

/**
 * Преобразует дату в красиво читаемый вид, но только год, месяц, день
 *
 * @param d дата для преобразования
 * @return строка, представляющей дату
 */
@SuppressWarnings( "deprecation" )
public static String doOnlyDate(Date d)
{
String toOut = "";
toOut += (d.getDate()>9?d.getDate():"0"+d.getDate()) + ".";
toOut += (d.getMonth()+1>9?d.getMonth()+1:"0"+(d.getMonth()+1)) + ".";
toOut += (d.getYear()+1900);
return toOut;
}

```

```

/**
 * Проверить имя на валидность
 *
 * @param name имя для проверки
 * @return имя, если проверка прошла успешно, иначе исключение
 *
 * @throws InvalidNameException
 */
public static String checkName4Salon(String name)
{
String controlStr = "ABCDEFGHIJKLMNOPQRSTUVWXYZ01234567890
zyxwvutsrqponmlkjihgfedcba_-, ";
for(int i = 0 ; i < name.length(); ++i)
if(controlStr.contains("" + name.charAt(i)) == false)
throw new InvalidNameException("You can only enter alphabetic characters, numbers and
characters \"_\", \"-\", \"'\", \". You inputted: \"" + name + "\". ", "" + name.charAt(i));
return name;
}
}

```

Employee.java


```

package lab;

import lab.*;

import java.lang.*;
import java.util.*;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;
import javax.persistence.*;

/**
 * Сотрудник салона красоты
 */
@Entity
@Table(name="EMPLOYEE")
public class Employee extends Person
{
    @Column(name="ACTIVE")
    private boolean ACTIVE;

    //CommentLink ste1
    //@OneToMany(fetch = FetchType.LAZY, mappedBy = "buffE", cascade = CascadeType.ALL)
    //private List<ServiceType> permittedServices;

    @ManyToMany(fetch = FetchType.LAZY, cascade = CascadeType.ALL)
    @JoinTable (name="EMPLOYEE_SERVICETYPE",
    joinColumns=@JoinColumn (name="M2M_EMPLOYEE_ID"),
    inverseJoinColumns=@JoinColumn(name="M2M_SERVICETYPE_ID"))
    private Set<ServiceType> permittedServices;

    @OneToOne (/*optional=false, */mappedBy="whoDo")
    private Service buffService;

    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "SALON_ID")
    private Salon salon;

    public Employee(){init4Const();}

    public Employee(String firstName, String secondName, boolean gender, String passport,
    Date birthday, Set<ServiceType> permittedServices, Salon salon)

```

```

{
    super(firstName, secondName, gender, passport, birthday);
    init4Const();
    this.ACTIVE = true;

    if(permitedServices != null)
    for(ServiceType st : permittedServices)
    this.addService(st);

    this.salon = salon;
}

private void init4Const()
{
    permittedServices = new HashSet<ServiceType>();
}

/**
 * Проверить активен ли сотрудник
 *
 * @return true=активный, false=неактивный
 */
public boolean isActive()
{
    return this.ACTIVE;
}

/**
 * Установить активность сотрудника
 *
 * @param a true=активен, false=неактивен
 */
public void active(boolean a)
{
    this.ACTIVE = a;
}

/**
 * Показывает в салоне услуги, которые сотрудник может оказывать
 */
public synchronized Set<ServiceType> getMasteredServices()
{
    return new HashSet<ServiceType>(permitedServices);
}

/**

```

```
* Теперь, если сотрудник компетентен в оказании услуги what, добавьте её к списку  
услуг, которые может оказывать сотрудник  
*/
```

```
public synchronized void addService(ServiceType what)  
{  
    //need to link with ServiceType.addServiceTypeMaster(...)  
    //6JlArO 3DECb PEKYPCu9 HE nPEDEJl!  
    if(!permittedServices.contains(what))  
    {  
        permittedServices.add(what);  
        what.addServiceTypeMaster(this);  
    }  
}
```

```
/**
```

```
* Теперь, если сотрудник компетентен в оказании услуги what, добавьте её к списку  
услуг, которые может оказывать сотрудник  
*/
```

```
public synchronized void addService(Set<ServiceType> sts, int[] which)  
{  
    int i, j, target;  
    for(i = 0; i < which.length; ++i)  
    {  
        target = which[i];  
        j = 1;  
        for(ServiceType _st : sts)  
        {  
            if(j == target)  
            {  
                this.addService(_st);  
                break;  
            }  
            ++j;  
        }  
    }  
}
```

```
/**
```

```
* Запрещает сотруднику оказывать услугу what  
*/
```

```
public synchronized void forbidService(ServiceType what)  
{  
    //need to link with ServiceType.removeServiceTypeMaster(...)  
    if(/*=(*/ permittedServices.contains(what) /*)=*/)  
    {  
        permittedServices.remove(what);  
        what.removeServiceTypeMaster(this);  
    }  
}
```

```

}

}

/**
 * Запретить сотруднику оказывать любые сервисы
 *
 */
public synchronized void forbidAllServices()
{
    permittedServices.clear();
}

public String toString()
{
    StringBuilder s = new StringBuilder();
    synchronized(this)
    {
        for(ServiceType st : this.permittedServices)
            s.append(st.getID() + " ");
    }
    return "Employee " + this.getID() + ": " + this.getName() + (ACTIVE==true?"", ":", not") +
    "active" + ". Can conduct services: { " + s + " }";
}

@Override
public boolean equals(Object otherObj)
{
    if(super.equals(otherObj))
    {
        synchronized(this)
        {
            Employee other = (Employee)otherObj;
            return this.ACTIVE == other.ACTIVE
            && this.permittedServices.equals(other.permittedServices)
            && this.salon.equals(other.salon);
        }
    }
    return false;
}
}

```

Client.java

```

package lab;

import lab.*;

```

```

import java.lang.*;
import java.util.*;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;
import javax.persistence.OneToOne;
import javax.persistence.ManyToOne;
import javax.persistence.JoinColumn;
import javax.persistence.FetchType;
import javax.persistence.*;

/**
 * Клиент салона красоты
 */
@Entity
@Table(name="CLIENT")
public class Client extends Person
{
    @Column(name="PRIORITY")
    /*Чем меньше число >= 0, тем более приоритетнее*/
    private int priority;

    @Column(name="BANNED")
    private boolean BANNED;

    @OneToOne (/*optional=false, */mappedBy="client")
    private Service buffService;

    @ManyToOne(optional=false, fetch = FetchType.LAZY)
    @JoinColumn(name = "SALON_ID")
    private Salon salon;

    public Client(){ }

    public Client(String firstName, String secondName, boolean gender, String passport,
    Date birthday, int priority, Salon salon)
    {
        super(firstName, secondName, gender, passport, birthday);
        //this.priority = priority;
        this.setPriority(priority);
        this.BANNED = false;
    }

```

```
this.salon = salon;
}

/**
 * Заблокировать клиента
 *
 * @param state true=заблокировать, false=разблокировать
 */
public void ban(boolean state)
{
    this.BANNED = state;
}

/**
 * Проверить заблокирован ли клиент
 *
 * @return true=заблокирован, false=незаблокирован
 */
public boolean isBanned()
{
    return this.BANNED;
}

/**
 * Получить приоритет клиента
 * Чем меньше число >= 0, тем более приоритетнее
 *
 * @return
 */
public int getPriority()
{
    return this.priority;
}

/**
 * Установить приоритет клиента
 * Чем меньше число >= 0, тем более приоритетнее, иначе исключение
 *
 * @param val приоритет
 * @throws NumberFormatException
 */
public void setPriority(int val) throws NumberFormatException
{
    if(val < 0)
        throw new NumberFormatException("Number must be >= 0. You inputted \"" + val + "\".");
    this.priority = val;
}
```

```

public String toString()
{
return "Client " + this.getID() + ": " + this.getName() + ", passport = " + this.getPassport() +
", priority: " + priority + (BANNED==false?", not ":", ") + "banned";
}

public boolean equals(Object otherObj)
{
if(super.equals(otherObj))
{
Client other = (Client)otherObj;
return this.priority == other.priority && this.salon.equals(other.salon);
}
return false;
}
}

```

Salon.java

```

package lab;

import lab.*;

import java.lang.*;
import java.util.*;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;
import javax.persistence.*;

/**
 * Сервис. Запись о том, какой сотрудник какому клиенту какую оказывал услугу, пере-
 численную в ServiceType, и за сколько
 */
@Entity
@Table(name="SERVICE")
public class Service
{
@Id
@GeneratedValue(strategy=GenerationType.IDENTITY)

```

```
@Column(name="SERVICEID")
private int serviceID;
```

```
@Column(name="DATEBEGIN")
private long dateBeginl;
```

```
@Column(name="CASHREWARD")
private double cashReward;
```

```
@Column(name="THENPRICE")
private double thenPrice;
```

```
@Column(name="REL")
private boolean REL;
```

```
@OneToOne (optional=false, cascade=CascadeType.ALL)
@JoinColumn (name="SERVICETYPE_ID")
private ServiceType serviceType;
```

```
@OneToOne (optional=false, cascade=CascadeType.ALL)
@JoinColumn (name="CLIENT_ID")
private Client client;
```

```
@OneToOne (optional=false, cascade=CascadeType.ALL)
@JoinColumn (name="EMPLOYEE_ID")
private Employee whoDo;
```

```
@ManyToOne(fetch = FetchType.LAZY)
@JoinColumn(name = "SALON_ID")
private Salon salon;
```

```
public Service(){}
```

```
public Service(ServiceType serviceType, Date dateBegin, Employee whoDo, double
cashReward, Client client, double thenPrice, Salon forWhichSalon)
{
    if(cashReward > thenPrice || thenPrice < 0 || cashReward < 0)
        throw new NumberFormatException("thenPrice must be more or equals then cashReward and
        positive: cashReward = \"\" + cashReward + "\", thenPrice = \"\" + thenPrice + "\".");
    this.serviceType = serviceType;
    this.dateBeginl = dateBegin.getTime();
    this.whoDo = whoDo;
    this.cashReward = cashReward;
    this.client = client;
    this.thenPrice = thenPrice;
    this.REL = true;
    this.salon = forWhichSalon;
```



```
}
```

```
public Service(ServiceType serviceType, Date dateBegin, Employee whoDo, Client client,  
Salon forWhichSalon)
```

```
{
```

```
this.serviceType = serviceType;
```

```
this.dateBeginI = dateBegin.getTime();
```

```
this.whoDo = whoDo;
```

```
this.cashReward = serviceType.getCurrentPrice() * serviceType.getPercent() * 0.01;
```

```
this.client = client;
```

```
this.thenPrice = serviceType.getCurrentPrice();
```

```
this.REL = true;
```

```
this.salon = forWhichSalon;
```

```
}
```

```
/**
```

```
 * Получить ID сервиса
```

```
 *
```

```
 * @return ID сервиса
```

```
 */
```

```
public int getID()
```

```
{
```

```
return this.serviceID;
```

```
}
```

```
/**
```

```
 * Получить дату начала сервиса
```

```
 *
```

```
 * @return дата начала сервиса
```

```
 */
```

```
public Date getDateBegin()
```

```
{
```

```
return new Date(dateBeginI);
```

```
}
```

```
/**
```

```
 * Получить сумму, которую заплатил клиент
```

```
 *
```

```
 * @return сумма, которую заплатил клиент
```

```
 */
```

```
public double getPrice()
```

```
{
```

```
return this.thenPrice;
```

```
}
```

```
/**
```

```
 * Получить тип сервиса
```

```

*
* @return
*/
public ServiceType getServiceType()
{
return this.serviceType;
}

/**
* Возвращает работника, который оказывал услугу
*/
public Employee getEmployee()
{
return whoDo;
}

/**
* Возвращает сумму, которую получил сотрудник после оказания услуги
*/
public double getCashReward()
{
return cashReward;
}

/**
* Получить клиента, которому оказывали услугу
*/
public Client getClient()
{
return client;
}

/**
* Возможно эта запись создана с ошибкой или ещё какая-нибудь причина, по которой
она не должна считаться
* Если true, то будет учитываться и показываться, если false, то эта запись не в счёт
*/
public boolean isRELEVANT()
{
return this.REL;
}

/**
* Возможно эта запись создана с ошибкой или ещё какая-нибудь причина, по которой
она не должна считаться
* Установите false, чтобы эта запись была не в счёт. Запись НЕ удалится из базы дан-
ных, лишь скроется

```

```

*/
public void setRELEVANT(boolean rel)
{
    this.REL = rel;
}

public String toString()
{
    return "Deal " + this.getID() + ": " + "The worker " + this.getEmployee().getName() + "
rendered a service id=" + this.getServiceType().getID() + " to the client " +
this.getClient().getName() + " on " + this.getDateBegin() + ". The client paid " +
this.getPrice();
}

```

@Override

```

public boolean equals(Object otherObj)
{
    if (otherObj == this) return true;
    if (otherObj == null) return false;
    if( this.getClass() != otherObj.getClass() ) return false;
    Service other = (Service)otherObj;
    return this.dateBeginI == other.dateBeginI
    && this.cashReward == other.cashReward
    && this.thenPrice == other.thenPrice
    && this.REL == other.REL
    && this.serviceType.equals(other.serviceType)
    && this.whoDo.equals(other.whoDo)
    && this.client.equals(other.client)
    && this.salon.equals(other.salon);
}
}

```

ServiceType.java

```

package lab;

import lab.*;

import java.lang.*;
import java.util.*;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;

```

```

import javax.persistence.*;

/**
 * Тип сервиса (специализация)
 */
@Entity
@Table(name="SERVICETYPE")
public class ServiceType
{
    @Id
    @GeneratedValue(strategy=GenerationType.IDENTITY)
    @Column(name="SERVICETYPEID")
    private int serviceTypeID;

    @Column(name="DESCRIPTION")
    private String description;

    @Column(name="CURRENTPRICE")
    private double currentPrice;

    /*Процент сотрудникам за оказание услуги. Число от 0 до 100.*/
    @Column(name="PERCENTAGETOEMPLOYEE")
    private float percentageToEmployee;

    @Column(name="RELEVANT")
    private boolean RELEVANT;

    //CommentLink ste1
    // @ManyToOne(*targetEntity = Employee.class, */optional=false, fetch = FetchType.LAZY)
    // @JoinColumn(name = "EMPLOYEE_ID")
    // private Employee buffE;

    //Thanks, hiber_nuts, for making me write extra code. YPOD
    @ManyToMany(fetch = FetchType.LAZY, cascade = CascadeType.ALL)
    @JoinTable(name="EMPLOYEE_SERVICETYPE",
        joinColumns=@JoinColumn(name="M2M_SERVICETYPE_ID"),
        inverseJoinColumns=@JoinColumn(name="M2M_EMPLOYEE_ID"))
    private Set<Employee> whoMastered;

    @OneToOne (*optional=false, */mappedBy="serviceType")
    private Service buffService;

    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "SALON_ID")
    private Salon salon;

```

```
public ServiceType(){init4Const();}
```

```
public ServiceType(String description, double price, float employeePercentage, Salon
forWhichSalon)
{
    init4Const();
    this.description = description;
    //this.currentPrice = price;
    this.setPrice(currentPrice);
    this.RELEVANT = true;
    //this.percentageToEmployee = employeePercentage;
    this.setPercent(employeePercentage);
    this.salon = forWhichSalon;
}
```

```
private void init4Const()
{
    whoMastered = new HashSet<Employee>();
}
```

```
/**
 * Получить описание типа сервиса
 *
 * @return
 */
```

```
public String getDescription()
{
    return this.description;
}
```

```
/**
 * Получить список сотрудников, которые могут оказывать данный сервис
 *
 * @return список сотрудников, которые могут оказывать данный сервис
 */
```

```
public synchronized Set<Employee> getWhoMasteredThisServiceTypes()
{
    return new HashSet<Employee>(whoMastered);
}
```

```
/**
 * Поменять описание типа сервиса
 *
 * @param desc новое описание
 */
```

```
public void changeDescription(String desc)
{
}
```

```

this.description = desc;
}

/**
 * Получить ID сервиса
 *
 * @return ID сервиса
 */
public int getID()
{
return this.serviceTypeID;
}

/**
 * Получить текущую стоимость услуги
 *
 * @return текущая стоимость услуги
 */
public double getCurrentPrice()
{
return this.currentPrice;
}

/**
 * Установить цену типа сервиса
 *
 * @param price новая цена
 */
public void setPrice(double price)
{
if(price < 0)
throw new NumberFormatException("price must be >= 0. You inputted: \"" + price + "\".");
this.currentPrice = price;
}

/**
 * Проверить, актуален ли данный тип сервиса
 *
 * @return true=актуален, false=неактуален
 */
public boolean isRelevant()
{
return this.RELEVANT;
}

/**
 * Установить актуальность сервиса

```

```

*
* @param rel true=актуален, false=неактуален
*/
public void relevant(boolean rel)
{
this.RELEVANT = rel;
}

/**
* Посмотреть процент, который начисляется сотруднику, от суммы стоимости услуги
* Число от 0 до 100
*/
public float getPercent()
{
return this.percentageToEmployee;
}

/**
* Установить процент, который начисляется сотруднику, от суммы стоимости услуги
* Число от 0 до 100, иначе исключение
*
* @param employeePercent процент, который начисляется сотруднику, от суммы стоимости услуги
* @throws NumberFormatException
*/
public void setPercent(float employeePercent)
{
if(employeePercent > 100 || employeePercent < 0)
throw new NumberFormatException("employeePercent must be 0 <= employeePercent <= 100. You inputted \" + employeePercent + "\".");
this.percentageToEmployee = employeePercent;
}

/**
* Добавить сотрудника, который может оказывать данный сервис
*
* @param who сотрудник, который может оказывать данный сервис
*/
public void addServiceTypeMaster(Employee who)
{
//need to link with Employee.addService(...)
if(!whoMastered.contains(who))
{
whoMastered.add(who);
who.addService(this);
}
}

```

```

/**
 * Убрать сотрудника, который раньше мог оказывать данный сервис
 *
 * @param who сотрудник, который раньше мог оказывать данный сервис
 */
public void removeServiceTypeMaster(Employee who)
{
    //KAK Tbl DOKATuJIC9 DO TAKOrO?!
    //need to link with Employee.forbidService(...)
    if(/*=(*/ whoMastered.contains(who) /*)=*/)
    {
        whoMastered.remove(who);
        who.forbidService(this);
    }
}

public String toString()
{
    StringBuilder whoMastered_strOut = new StringBuilder();
    for(Employee e : whoMastered)
        whoMastered_strOut.append(e.getID() + " ");
    return "Service " + this.getID() + ": " + this.getDescription() + ". It costs " + currentPrice + ".
    This service can be provided by employees: " + whoMastered_strOut.toString() + ". ";
}

@Override
public boolean equals(Object otherObj)
{
    if (otherObj == this) return true;
    if (otherObj == null) return false;
    if (this.getClass() != otherObj.getClass() ) return false;
    ServiceType other = (ServiceType)otherObj;
    return this.currentPrice == other.currentPrice
    && this.percentageToEmployee == other.percentageToEmployee
    && this.RELEVANT == other.RELEVANT
    && this.description.equals(other.description)
    && this.whoMastered.equals(other.whoMastered)
    && this.salon.equals(other.salon);
}
}

```

Person.java

```
package lab;
```



```

import lab.*;

import java.lang.*;
import java.util.*;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;
//import javax.persistence.Inheritance;
//import javax.persistence.InheritanceType;
import javax.persistence.MappedSuperclass;

@Entity
@Inheritance(strategy = InheritanceType.TABLE_PER_CLASS)
/**
 * Класс человек. От него наследуются Employee и Client
 */
@MappedSuperclass
@Table(name="PERSON")
public class Person
{
    @Id
    @GeneratedValue(strategy=GenerationType.IDENTITY)
    @Column(name="PERSONID")
    private int personID;

    @Column(name="FIRSTNAME")
    private String firstName;

    @Column(name="SECONDNAME")
    private String secondName;

    @Column(name="BIRTHDAY")
    private long birthday;

    //private Date birthday;

    @Column(name="PASSPORT")
    private String passport;

    /*1-man, 0-woman*/
    @Column(name="GENDER")

```

```
private int gender;
```

```
public Person(){} 
```

```
public Person(String firstName, String secondName, boolean gender, String passport,
Date birthday)
```

```
{
this.firstName = firstName;
this.secondName = secondName;
this.gender = gender==true?1:0;
//this.birthday = birthday;
this.birthdayl = birthday.getTime();
this.personID = personID;
this.passport = passport;
}
```

```
/**
```

```
 * Получить ID человека
```

```
 *
```

```
 * @return ID человека
```

```
 */
```

```
public int getID()
```

```
{
return personID;
}
```

```
/**
```

```
 * Получить полное имя человека
```

```
 *
```

```
 * @return полное имя человека
```

```
 */
```

```
public String getName()
```

```
{
return secondName + " " + firstName;
}
```

```
/**
```

```
 * Получить только имя человека
```

```
 *
```

```
 * @return имя человека
```

```
 */
```

```
public String getFirstName()
```

```
{
return firstName;
}
```

```
/**
```

```

* Получить фамилию человека
*
* @return фамилия человека
*/
public String getSecondName()
{
return secondName;
}

/**
* Установить имя человека
*
* @param firstName имя
* @param secondName фамилия
*/
public void setName(String firstName, String secondName)
{
this.firstName = firstName;
this.secondName = secondName;
}

/**
* Установить паспорт человека
*
* @param passport паспорт человека
*/
public void setPassport(String passport)
{
this.passport = passport;
}

/**
* Установить пол человека
*
* @param s true=man, false=female
*/
public void setGender(boolean s)
{
this.gender = (s==true?1:0);
}

/**
* Задать дату рождения человека
*
* @param d дата рождения человека
*/
public void setBirthday(Date d)

```

```
{  
this.birthdayl = d.getTime();  
}
```

```
/**  
 * Получить дату рождения  
 *  
 * @return дата рождения  
 */  
public Date getBirthday()  
{  
return new Date(birthdayl);  
}
```

```
/**  
 * Получить паспорт человека  
 *  
 * @return паспорт человека  
 */  
public String getPassport()  
{  
return passport;  
}
```

```
/**  
 * Получить пол человека  
 *  
 * @return true=man, false=female  
 */  
public boolean getGender()  
{  
return gender==1?true:false;  
}
```

```
public String toString()  
{  
return personID + ": " + firstName + ", " + secondName + " is " +  
(gender==1?"male":"female") + ", passport=" + passport + ", birthday is " + new  
Date(birthdayl);  
}
```

```
@Override  
public boolean equals(Object otherObj)  
{  
if (otherObj == this) return true;  
if (otherObj == null) return false;  
if (this.getClass() != otherObj.getClass() ) return false;
```

```

Person other = (Person)otherObj;

return this.birthdayI == other.birthdayI
&& this.gender == other.gender
&& this.firstName.equals(other.firstName)
&& this.secondName.equals(other.secondName)
&& this.passport.equals(other.passport);
}
}

```

GreateGenerator.java

```

package lab;

import lab.*;

import java.lang.*;
import java.util.*;

/**
 * Генератор салона. Использовалась модуль mimesis для Python3
 */
public class GreatGenerator
{
    private static Comparator<Service> servicesDateComporator;

    static
    {
        servicesDateComporator = (Service o1, Service o2) ->
        (o2.getDateBegin().compareTo(o1.getDateBegin()));
    }

    /**
     * Получить компаратор для сравнения сервисов по дате
     */
    * @return компаратор для сравнения сервисов по дате
    */
    public static Comparator<Service> getServicesDateComporator()
    {
        return servicesDateComporator;
    }

    public static void main(String[] ars)
    {
        Salon salon = new Salon("Salon Harry Dubua face");
        makeSalon(salon);
    }
}

```

```

System.out.println(salon);
}

/**
 * Создать салон с уже большим количеством записей
 *
 * @param salon
 */
public static void makeSalon(Salon salon)
{
    ...
}

/**
 * Получить элемент из множества с индексом which
 *
 * @param seterino множество
 * @param which индекс получаемого элемента
 * @return элемент из множества с индексом which
 */
public static Object getElFromSet(Set<? extends Object> seterino, int which)
{
    int j;
    j = 0;
    for(Object obj : seterino)
    {
        if(j == which)
            return obj;
        ++j;
    }
    return null;
}
}

```

HibHundler.java

```

package lab;

import lab.*;

import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.cfg.Configuration;

import java.lang.*;
import java.util.*;

```

```

import java.io.*;

/**
 * Обработчик для hibernate
 */
public class HibHundler
{
    private static SessionFactory factory;
    private static Session session;
    static
    {
        //initFactoryAndSession();
    }

    /**
     * Загрузить салон из БД
     *
     * @return загруженный салон
     */
    public static Salon loadSalon()
    {
        int sad_i = 1;
        Salon loadedSalon = null;
        do
        {
            loadedSalon = session.get(Salon.class, sad_i++);
        } while (loadedSalon == null);
        return loadedSalon;
    }

    /**
     * Сохранить салон в БД
     *
     * @param salon сохраняемый салон
     */
    public static void saveSalon(Salon salon)
    {
        session.saveOrUpdate(salon);
    }

    /**
     * Сохранить сущность в БД
     *
     * @param obj сущность
     */
    public static void saveObject(Object obj)

```

```

{
    session.saveOrUpdate(obj);
}

/**
 * Проинициализировать factory и session для hibernate
 *
 */
public static void initFactoryAndSession()
{
    factory = new Configuration()
        .configure(HibHundler.class.getResource("/hibernate.cfg.xml"))
        /*.configure("hibernate.cfg.xml")*/
        .addAnnotatedClass(Person.class)
        .addAnnotatedClass(Client.class)
        .addAnnotatedClass(Employee.class)
        .addAnnotatedClass(ServiceType.class)
        .addAnnotatedClass(Service.class)
        .addAnnotatedClass(Salon.class)
        .buildSessionFactory();
    session = factory.getCurrentSession();
    session.beginTransaction();
}

/**
 * Закрывать factory и session для hibernate
 *
 */
public static void closeFactoryAndSession()
{
    session.getTransaction().commit();
    factory.close();
}
}

```

InvalidNameException.java

```

package lab;

import lab.*;

import java.lang.*;
import java.util.*;

/**
 * Класс исключения при неправильном вводе имени

```



```

*/
*/
public class InvalidNameException extends IllegalArgumentException
{
    String kaka;

    public InvalidNameException(String message, String InvalidSymbol)
    {
        super(message);
        this.kaka = "\"" + InvalidSymbol + "\"";
    }

    /**
     * Получить символ, который вызвал исключение
     *
     * @return символ, который вызвал исключение
     */
    public String getInvalidSymbol()
    {
        return this.kaka;
    }
}

```

ReportMaker.java

```

package lab;

import java.lang.*;
import java.util.*;

import lab.*;

import java.util.List;

import java.io.FileOutputStream;
import java.util.Date;
import com.itextpdf.text.*;
import com.itextpdf.text.pdf.*;

import org.apache.log4j.Logger;

/**
 * Класс, отвечающий за генерацию отчётов PDF
 *
 */
public class ReportMaker

```

```

{
private static final Logger log = Logger.getLogger(ReportMaker.class);

//private String FILE = "./Report.pdf";
private String FILE;
private static Font t_font = new Font(Font.FontFamily.TIMES_ROMAN, 18, Font.NORMAL);
private static Font c_font = new Font(Font.FontFamily.TIMES_ROMAN, 18, Font.NORMAL);
private Document document;

private Salon salon;
private int D_W_M;
private Date d;
private Set<Service> ss;

/**
 * Сделать отчёт PDF
 *
 * @param salon салон, отчёт которого делается
 * @param ss множество сервисов, которые будут отражены в отчёте
 * @param D_W_M день=0/неделя=1/месяц=2
 * @param d дата соответствующая D_W_M
 * @param path путь к файлу
 */
public synchronized static void makeReport(Salon salon, Set<Service> ss, int D_W_M,
Date d, String path)
{
    new ReportMaker(salon, ss, D_W_M, d, path);

    log.info("OT4ET PDF rOTOB");
}

private ReportMaker(Salon salon, Set<Service> ss, int D_W_M, Date d, String path)
{
    this.salon = salon;
    this.D_W_M = D_W_M;
    this.d = d;
    this.ss = ss;
    this.FILE = path;

    try
    {
        //document = new Document();
        document = new Document(PageSize.A4.rotate());
        PdfWriter.getInstance(document, new FileOutputStream(FILE));
        document.open();
        addMetaData();
        addTitle();
    }
}

```

```
dododo();  
}  
catch(Exception e)  
{  
e.printStackTrace();  
}  
}
```

```
private void addMetaData()  
{  
document.addTitle("Report");  
document.addSubject("Using iText");  
document.addAuthor("9");  
document.addCreator("DA, DA 9");  
}
```

```
private void addTitle()  
{  
Paragraph preface = new Paragraph();  
addEmptyLine(preface, 1);  
preface.add(new Paragraph("Title", t_font));  
addEmptyLine(preface, 1);  
String deFirst = "Report generated by " + System.getProperty("user.name") + ", " + new  
Date() + " \n" + "Services\' days: " + takeDates();  
preface.add(new Paragraph(deFirst, c_font));  
try  
{  
document.add(preface);  
}  
catch(Exception e)  
{  
e.printStackTrace();  
}  
document.newPage();  
}
```

```
private void dododo()  
{  
Anchor anchor = new Anchor("Service", t_font);  
anchor.setName("Services");  
Paragraph ph = new Paragraph(anchor);  
Chapter chapter = new Chapter(ph, 1);  
//Section section = chapter.addSection(ph);
```

```
PdfPTable table = new PdfPTable(7);  
table.setWidthPercentage(100);  
table.setSpacingBefore(20f);
```

```

table.setSpacingAfter(20f);
PdfPCell pfpc;

pfpc = new PdfPCell(new Phrase("ID"));
pfpc.setHorizontalAlignment(Element.ALIGN_CENTER);
table.addCell(pfpc);

pfpc = new PdfPCell(new Phrase("Service type"));
pfpc.setHorizontalAlignment(Element.ALIGN_CENTER);
table.addCell(pfpc);

pfpc = new PdfPCell(new Phrase("Date"));
pfpc.setHorizontalAlignment(Element.ALIGN_CENTER);
table.addCell(pfpc);

pfpc = new PdfPCell(new Phrase("Employee"));
pfpc.setHorizontalAlignment(Element.ALIGN_CENTER);
table.addCell(pfpc);

pfpc = new PdfPCell(new Phrase("Employee's cut"));
pfpc.setHorizontalAlignment(Element.ALIGN_CENTER);
table.addCell(pfpc);

pfpc = new PdfPCell(new Phrase("Client"));
pfpc.setHorizontalAlignment(Element.ALIGN_CENTER);
table.addCell(pfpc);

pfpc = new PdfPCell(new Phrase("Price"));
pfpc.setHorizontalAlignment(Element.ALIGN_CENTER);
table.addCell(pfpc);

for(Service s : ss)
{
if(s.isRELEVANT() != false)
{
table.addCell("" + s.getID());
table.addCell(s.getServiceType().getDescription() + " - " + s.getServiceType().getID());
table.addCell(Salon.doOnlyDate(s.getDateBegin()));
table.addCell(s.getEmployee().getName() + " - " + s.getEmployee().getID());
table.addCell("" + s.getCashReward());
table.addCell(s.getClient().getName() + " - " + s.getClient().getID());
table.addCell("" + s.getPrice());
}
}
table.setHeaderRows(1);
chapter.add(table);
try

```

```

{
document.add(chapter);
document.newPage();
}
catch(Exception e)
{
e.printStackTrace();
}
finally
{
doClose();
}
}

```

```

@SuppressWarnings( "deprecation" )

```

```

private String takeDates()

```

```

{
String res = "";
if(D_W_M == 0)
res = Salon.doOnlyDate(d);
else if(D_W_M == 1)
{
Date[] dts = Salon.getWeekKnowsDay(d);
res = Salon.doOnlyDate(dts[0]) + " - " + Salon.doOnlyDate(dts[6]);
}
else if(D_W_M == 2)
{
Date be = new Date(d.getTime()); be.setDate(1);
Date en = new Date(d.getTime()); en.setDate(30); // DA 6Jl*! 30 uJlu 31? uJlu 29?
res = Salon.doOnlyDate(be) + " - " + Salon.doOnlyDate(en);
}
return res;
}

```

```

private static void addEmptyLine(Paragraph paragraph, int number)

```

```

{
for (int i = 0; i < number; ++i)
{
paragraph.add(new Paragraph(" "));
}
}

```

```

private boolean isOpen()

```

```

{
return document.isOpen();
}

```

```
private void doOpen()
{
    document.open();
}
```

```
private void doClose()
{
    document.close();
}
}
```

AddClient.java

```
package labgui;
```

```
import java.lang.*;
import java.util.*;
import java.awt.*;
import javax.swing.*;
import javax.swing.table.*;
import java.awt.event.*;
```

```
import lab.*;
import labgui.*;
```

```
import org.apache.log4j.Logger;
```

```
/**
 * Форма для добавления клиента
 */
```

```
class AddClient extends JDialog
```

```
{
    private static final Logger log = Logger.getLogger(AddClient.class);
```

```
    private SalonGUI salonGUI;
    private Salon salon;
```

```
    private final int DEFAULT_WIDTH = 500;
    private final int DEFAULT_HEIGHT = 750;
```

```
    private JPanel mainPanel;
    private JButton doneButton;
```

```
    private JTextField firstNameText;
    private JTextField secondNameText;
```

```

private DateChooser dateChooser;
private JTextField passportText;
private boolean gender;
private boolean state;
private JTextField priorityText;

public AddClient(JFrame owner)
{
    super(owner, "Add client", false);

    log.info("Creating new AddClient frame...");

    mainPanel = new JPanel();
    /*
    firstName: []
    secondName: []
    birthday: []
    passport: []
    gender: *
    BANNED: *
    priority: []
    [done]
    */
    mainPanel.setLayout(new GridLayout(8, 0, 15, 15));
    this.add(mainPanel);
    updateAdd();

    this.setLocationRelativeTo(owner);
    pack();

    log.info("AddClient frame was created.");
}

/**
 * Обновить содержимое формы
 */
public void updateAdd()
{
    mainPanel.removeAll();

    salonGUI = SalonGUI.getSalonGUI();
    salon = salonGUI.getSalon();

    //firstName
    JPanel fnp = new JPanel();
    fnp.add(new JLabel("First name: "));

```

```
firstNameText = new JTextField("", 20);
fnp.add(firstNameText);
mainPanel.add(fnp);
```

```
//secondName
```

```
JPanel cnp = new JPanel();
cnp.add(new JLabel("Second name: "));
secondNameText = new JTextField("", 20);
cnp.add(secondNameText);
mainPanel.add(cnp);
```

```
//birthday
```

```
JPanel bp = new JPanel();
bp.add(new JLabel("Birthday: "));
dateChooser = new DateChooser();
bp.add(dateChooser);
mainPanel.add(bp);
```

```
//passport
```

```
JPanel pp = new JPanel();
pp.add(new JLabel("Passport: "));
passportText = new JTextField("", 20);
pp.add(passportText);
mainPanel.add(pp);
```

```
//gender
```

```
JPanel gp = new JPanel();

gender = true;
JPanel gender4Ratio = new JPanel();
JRadioButton genderMaleRatio = new JRadioButton("Male");
genderMaleRatio.setSelected(true);
JRadioButton genderFemaleRatio = new JRadioButton("Female");
ActionListener genderActionListener = ae ->
{
    if(genderMaleRatio.isSelected())
        gender = true;
    else if(genderFemaleRatio.isSelected())
        gender = false;
};
genderMaleRatio.addActionListener(genderActionListener);
genderFemaleRatio.addActionListener(genderActionListener);
ButtonGroup genderRatioGroup = new ButtonGroup();
genderRatioGroup.add(genderMaleRatio);
genderRatioGroup.add(genderFemaleRatio);
gender4Ratio.add(genderMaleRatio);
gender4Ratio.add(genderFemaleRatio);
```



```
gp.add(new JLabel("Gender: "));
gp.add(gender4Ratio);
mainPanel.add(gp);
```

```
//BANNED
```

```
JPanel ap = new JPanel();
ap.add(new JLabel("Banned: "));
JCheckBox cb = new JCheckBox("is banned?");
state = false;
cb.setSelected(state);
cb.addItemListener(new ItemListener() {
    public void itemStateChanged(ItemEvent ie) {
        state = cb.isSelected();
    }
});
ap.add(cb);
mainPanel.add(ap);
```

```
//priority
```

```
JPanel prp = new JPanel();
prp.add(new JLabel("Priority: "));
priorityText = new JTextField("", 20);
prp.add(priorityText);
mainPanel.add(prp);
```

```
//Button
```

```
doneButton = new JButton("Done!");
doneButton.addActionListener(aeDone -> {addClient();});
mainPanel.add(doneButton);
```

```
mainPanel.updateUI();
}
```

```
public Dimension getPreferredSize()
{
    return new Dimension(DEFAULT_WIDTH, DEFAULT_HEIGHT);
}
```

```
@SuppressWarnings("deprecation")
private void addClient()
{
    log.info("Adding new client...");
}
```

```
String firstName = "";
String secondName = "";
Date d = new Date();
```

```
String passport = "";  
int priority = -1;
```

```
Client c;
```

```
try  
{  
    //names  
    firstName = Salon.checkName4Salon(firstNameText.getText());  
    secondName = Salon.checkName4Salon(secondNameText.getText());  
  
    //birthday  
    d = dateChooser.getDate();  
  
    //passport  
    passport = Salon.checkName4Salon(passportText.getText());  
  
    //priority  
    priority = Integer.parseInt( priorityText.getText() );  
    c = new Client(firstName, secondName, gender, passport, d, priority, salon);  
}  
catch(NumberFormatException ne)  
{  
    String msg = ne.getMessage();  
    if(msg.indexOf("\") != -1)  
        msg = msg.substring(msg.indexOf("\"), msg.lastIndexOf("\")+1);  
    JOptionPane.showMessageDialog(this, "You inputted wrong number: " + msg + ".", "Error",  
    JOptionPane.ERROR_MESSAGE);  
    return;  
}  
catch(InvalidNameException ie)  
{  
    JOptionPane.showMessageDialog(this, ie.getMessage() + "Bab symbol: " +  
    ie.getInvalidSymbol(), "Error", JOptionPane.ERROR_MESSAGE);  
    return;  
}  
catch(Exception ignor)  
{  
    log.error("Problem with adding client: " + ignor);  
    //ignor.printStackTrace();  
    return;  
}  
  
//BANNED  
c.ban(state);  
  
salon.addClient(c);
```

```

try
{
HibHundler.initFactoryAndSession();
HibHundler.saveObject(c);
HibHundler.closeFactoryAndSession();
}
catch(*Exception */Throwable ex)
{
log.error("Problem with save into DB client: " + c + ": " + ex);
//ex.printStackTrace();
JOptionPane.showMessageDialog(this, "Problem with save into DB");
return;
}

JOptionPane.showMessageDialog(this, "Client (id=" + c.getID() + ") was saved in Data Base");
salonGUI.refreshTablesOf_Clients();

log.info("New client added.");
}
}

```

AddEmployee.java

```

package labgui;

import java.lang.*;
import java.util.*;
import java.awt.*;
import javax.swing.*;
import javax.swing.table.*;
import java.awt.event.*;

import lab.*;
import labgui.*;

import org.apache.log4j.Logger;

/**
 * Форма для добавления сотрудника
 */
class AddEmployee extends JDialog
{
private static final Logger log = Logger.getLogger(AddEmployee.class);

```

```

private SalonGUI salonGUI;
private Salon salon;

private final int DEFAULT_WIDTH = 500;
private final int DEFAULT_HEIGHT = 700;

private JPanel mainPanel;
private JButton doneButton;

private JTextField firstNameText;
private JTextField secondNameText;
private DateChooser dateChooser;
private JTextField passportText;
private boolean gender;
private boolean state;
private JTextField permittedServicesText;

public AddEmployee(JFrame owner)
{
    super(owner, "Add employee", false);

    log.info("Creating new AddEmployee frame...");

    mainPanel = new JPanel();
    /*
    firstName: []
    secondName: []
    birthday: []
    passport: []
    gender: * *
    ACTIVE: *
    permittedServices: []
    [done]
    */
    mainPanel.setLayout(new GridLayout(8, 0, 15, 15));
    this.add(mainPanel);
    updateAdd();

    this.setLocationRelativeTo(owner);
    pack();

    log.info("AddEmployee frame was created.");
}

/**
 * Обновить содержимое формы
 */

```

```

*/
public void updateAdd()
{
    mainPanel.removeAll();

    salonGUI = SalonGUI.getSalonGUI();
    salon = salonGUI.getSalon();

    //firstName
    JPanel fnp = new JPanel();
    fnp.add(new JLabel("First name: "));
    firstNameText = new JTextField("", 20);
    fnp.add(firstNameText);
    mainPanel.add(fnp);

    //secondName
    JPanel cnp = new JPanel();
    cnp.add(new JLabel("Second name: "));
    secondNameText = new JTextField("", 20);
    cnp.add(secondNameText);
    mainPanel.add(cnp);

    //birthday
    JPanel bp = new JPanel();
    bp.add(new JLabel("Birthday: "));
    dateChooser = new DateChooser();
    bp.add(dateChooser);
    mainPanel.add(bp);

    //passport
    JPanel pp = new JPanel();
    pp.add(new JLabel("Passport: "));
    passportText = new JTextField("", 20);
    pp.add(passportText);
    mainPanel.add(pp);

    //gender
    JPanel gp = new JPanel();

    gender = true;
    JPanel gender4Ratio = new JPanel();
    JRadioButton genderMaleRatio = new JRadioButton("Male");
    genderMaleRatio.setSelected(true);
    JRadioButton genderFemaleRatio = new JRadioButton("Female");
    ActionListener genderActionListener = ae ->
    {
        if(genderMaleRatio.isSelected())

```

```

gender = true;
else if(genderFemaleRatio.isSelected())
gender = false;
};
genderMaleRatio.addActionListener(genderActionListener);
genderFemaleRatio.addActionListener(genderActionListener);
ButtonGroup genderRatioGroup = new ButtonGroup();
genderRatioGroup.add(genderMaleRatio);
genderRatioGroup.add(genderFemaleRatio);
gender4Ratio.add(genderMaleRatio);
gender4Ratio.add(genderFemaleRatio);

gp.add(new JLabel("Gender: "));
gp.add(gender4Ratio);
mainPanel.add(gp);

//ACTIVE
JPanel ap = new JPanel();
ap.add(new JLabel("State: "));
JCheckBox cb = new JCheckBox("relevant");
state = true;
cb.setSelected(state);
cb.addItemListener(new ItemListener() {
public void itemStateChanged(ItemEvent ie) {
state = cb.isSelected();
}
});
ap.add(cb);
mainPanel.add(ap);

//permittedServices
JPanel psp = new JPanel();
psp.add(new JLabel("Permitted services: "));
permittedServicesText = new JTextField("", 15);
psp.add(permittedServicesText);
mainPanel.add(psp);

//Button
doneButton = new JButton("Done!");
doneButton.addActionListener(aeDone -> {addEmployee();});
mainPanel.add(doneButton);

mainPanel.updateUI();
}

public Dimension getPreferredSize()
{

```

```
return new Dimension(DEFAULT_WIDTH, DEFAULT_HEIGHT);
}
```

```
@SuppressWarnings( "deprecation" )
```

```
private void addEmployee()
```

```
{
```

```
log.info("Adding new employee...");
```

```
String firstName = "";
```

```
String secondName = "";
```

```
Date d = new Date();
```

```
String passport = "";
```

```
String[] buffSs = null;
```

```
Set<ServiceType> sts = null;
```

```
Employee e;
```

```
try
```

```
{
```

```
//names
```

```
firstName = Salon.checkName4Salon(firstNameText.getText());
```

```
secondName = Salon.checkName4Salon(secondNameText.getText());
```

```
//birthday
```

```
d = dateChooser.getDate();
```

```
//passport
```

```
passport = Salon.checkName4Salon(passportText.getText());
```

```
//permittedServices
```

```
sts = new HashSet<ServiceType>();
```

```
if(!permittedServicesText.getText().trim().equals(""))
```

```
{
```

```
buffSs = permittedServicesText.getText().trim().split(" ");
```

```
int[] pss = new int[buffSs.length];
```

```
for(int i = 0; i < pss.length; ++i)
```

```
pss[i] = Integer.parseInt(buffSs[i]);
```

```
for(int i = 0; i < pss.length; ++i)
```

```
{
```

```
ServiceType buffST = salon.getServiceTypeID(pss[i]);
```

```
if(buffST == null)
```

```
throw new NumberFormatException("'" + pss[i]);
```

```
sts.add(buffST);
```

```
}
```

```
}
```

```
e = new Employee(firstName, secondName, gender, passport, d, sts, salon);
```

```

    }
    catch(NumberFormatException ne)
    {
        String msg = ne.getMessage();
        if(msg.indexOf("\"") != -1)
            msg = msg.substring(msg.indexOf("\""), msg.lastIndexOf("\"")+1);
        JOptionPane.showMessageDialog(this, "You inputted wrong number: " + msg + ".", "Error",
        JOptionPane.ERROR_MESSAGE);
        return;
    }
    catch(InvalidNameException ie)
    {
        JOptionPane.showMessageDialog(this, ie.getMessage() + "Bab symbol: " +
        ie.getInvalidSymbol(), "Error", JOptionPane.ERROR_MESSAGE);
        return;
    }
    catch(Exception ignor)
    {
        log.error("Problem with adding employee: " + ignor);
        //ignor.printStackTrace();
        return;
    }
    //ACTIVE
    e.active(state);

    salon.addEmployee(e);

    try
    {
        HibHundler.initFactoryAndSession();
        HibHundler.saveObject(e);
        HibHundler.closeFactoryAndSession();
    }
    catch(/*Exception */Throwable ex)
    {
        log.error("Problem with save into DB employee: " + e + ": " + ex);
        //ex.printStackTrace();
        JOptionPane.showMessageDialog(this, "Problem with save into DB");
        return;
    }

    JOptionPane.showMessageDialog(this, "Employee (id=" + e.getID() + ") was saved in Data
    Base");

    salonGUI.refreshTablesOf_Employees();

    log.info("New employee added.");

```



```
}  
}
```

AddService.java

```
package labgui;  
  
import java.lang.*;  
import java.util.*;  
import java.awt.*;  
import javax.swing.*;  
import javax.swing.table.*;  
import java.awt.event.*;  
  
import lab.*;  
import labgui.*;  
  
import org.apache.log4j.Logger;  
  
/**  
 * Форма для добавления сервиса  
 */  
class AddService extends JDialog  
{  
    private static final Logger log = Logger.getLogger(AddService.class);  
  
    private SalonGUI salonGUI;  
    private Salon salon;  
  
    private final int DEFAULT_WIDTH = 500;  
    private final int DEFAULT_HEIGHT = 700;  
  
    private JPanel mainPanel;  
    private JButton doneButton;  
  
    private JTextField clientCostText;  
    private JTextField employeeSalaryText;  
    private DateChooser dateChooser;  
  
    private int selectedServiceType;  
    private int selectedEmployee;  
    private int selectedClient;
```

```

private boolean state;

public AddService(JFrame owner)
{
    super(owner, "Add specialization", false);

    log.info("Creating new AddService frame...");

    mainPanel = new JPanel();
    /*
    what: [combo]
    whom: [combo]
    who: [combo]
    cost: []
    salary: []
    date: []
    REL: *
    [done]
    */
    mainPanel.setLayout(new GridLayout(8, 1, 15, 15));
    this.add(mainPanel);
    updateAdd();

    this.setLocationRelativeTo(owner);
    pack();

    log.info("AddService frame was created.");
}

/**
 * Обновить содержимое формы
 */
public void updateAdd()
{
    mainPanel.removeAll();

    salonGUI = SalonGUI.getSalonGUI();
    salon = salonGUI.getSalon();

    selectedServiceType = -1;
    selectedClient = -1;
    selectedEmployee = -1;
    clientCostText = new JTextField("", 20);
    employeeSalaryText = new JTextField("", 20);
    //==creatPlate==
    int i;

```

//ServiceTypes

```
JPanel stsp = new JPanel();
Set<ServiceType> sts = salon.getProvidedServices();
String[] serviceTypesCombo = new String[sts.size()];
i = 0;
for(ServiceType item : sts)
{
    serviceTypesCombo[i] = item.getDescription() + " - " + item.getID();
    ++i;
}
Arrays.sort(serviceTypesCombo, (String a, String b) -> {return a.compareTo(b);});
JComboBox serviceTypesComboBox = new JComboBox<String>(serviceTypesCombo);
ActionListener serviceTypeActionListener = new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        JComboBox box = (JComboBox)e.getSource();
        selectedServiceType = takeIDfromCombo((String)box.getSelectedItem());
        clientCostText.setText("" + salon.getServiceTypeByID(selectedServiceType).getCurrentPrice());
        setEmployeeSalaryText();
    }
};
serviceTypesComboBox.addActionListener(serviceTypeActionListener);
stsp.add(new JLabel("What: "));
stsp.add(serviceTypesComboBox);
mainPanel.add(stsp);
```

//Clients

```
JPanel csp = new JPanel();
Set<Client> cs = salon.getClients();
String[] clientsCombo = new String[cs.size()];
i = 0;
for(Client item : cs)
{
    clientsCombo[i] = item.getName() + " - " + item.getID();
    ++i;
}
Arrays.sort(clientsCombo, (String a, String b) -> {return a.compareTo(b);});
JComboBox clientsComboBox = new JComboBox<String>(clientsCombo);
ActionListener clientActionListener = new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        JComboBox box = (JComboBox)e.getSource();
        selectedClient = takeIDfromCombo((String)box.getSelectedItem());
    }
};
clientsComboBox.addActionListener(clientActionListener);
csp.add(new JLabel("Whom: "));
csp.add(clientsComboBox);
```

```

mainPanel.add(csp);

//Employees
JPanel esp = new JPanel();
Set<Employee> es = salon.getEmployees();
String[] employeesCombo = new String[es.size()];
i = 0;
for(Employee item : es)
{
employeesCombo[i] = item.getName() + " - " + item.getID();
++i;
}
Arrays.sort(employeesCombo, (String a, String b) -> {return a.compareTo(b);});
JComboBox employeesComboBox = new JComboBox<String>(employeesCombo);
ActionListener employeeActionListener = new ActionListener() {
public void actionPerformed(ActionEvent e) {
JComboBox box = (JComboBox)e.getSource();
selectedEmployee = takeIDfromCombo((String)box.getSelectedItem());
}
};
employeesComboBox.addActionListener(employeeActionListener);
esp.add(new JLabel("Who: "));
esp.add(employeesComboBox);
mainPanel.add(esp);

```

```

//cost
JPanel costp = new JPanel();
costp.add(new JLabel("Client cost: "));
clientCostText.addActionListener(eve ->
{
setEmployeeSalaryText();
});
costp.add(clientCostText);
mainPanel.add(costp);

```

```

//salary
JPanel sap = new JPanel();
sap.add(new JLabel("Employee's cut: "));
sap.add(employeeSalaryText);
mainPanel.add(sap);

```

```

//date
JPanel datep = new JPanel();
datep.add(new JLabel("Date: "));
dateChooser = new DateChooser();
datep.add(dateChooser);

```

```
mainPanel.add(datep);
```

```
//=====
```

```
//REL
```

```
JPanel rp = new JPanel();
```

```
rp.add(new JLabel("State: "));
```

```
JCheckBox cb = new JCheckBox("relevant");
```

```
state = true;
```

```
cb.setSelected(state);
```

```
cb.addItemListener(new ItemListener() {
```

```
public void itemStateChanged(ItemEvent ie) {
```

```
state = cb.isSelected();
```

```
}
```

```
});
```

```
rp.add(cb);
```

```
mainPanel.add(rp);
```

```
//Button
```

```
doneButton = new JButton("Done!");
```

```
doneButton.addActionListener(aeDone -> {addService();});
```

```
mainPanel.add(doneButton);
```

```
mainPanel.updateUI();
```

```
}
```

```
private static int takeIDfromCombo(String s)
```

```
{
```

```
int res;
```

```
int i = s.lastIndexOf(" - ");
```

```
String buffS = s.substring(i + 3, s.length());
```

```
res = Integer.parseInt(buffS);
```

```
return res;
```

```
}
```

```
private void setEmployeeSalaryText()
```

```
{
```

```
if(selectedServiceType != -1)
```

```
{
```

```
try
```

```
{
```

```
double d = Double.parseDouble(clientCostText.getText());
```

```
float percent = salon.getServiceTypeByID(selectedServiceType).getPercent();
```

```
employeeSalaryText.setText("" + d*percent*0.01);
```

```
}
```

```
catch(Exception ignor)
```

```
{
```

```
}  
}  
  
}
```

```
public Dimension getPreferredSize()  
{  
    return new Dimension(DEFAULT_WIDTH, DEFAULT_HEIGHT);  
}
```

```
@SuppressWarnings( "deprecation" )  
private void addService()  
{  
    log.info("Adding new service...");
```

```
if(selectedServiceType == -1 || selectedClient == -1 || selectedEmployee == -1)  
{  
    JOptionPane.showMessageDialog(this, "Please fill in all the fields.");  
    return;  
}
```

```
ServiceType st = null;  
Client c = null;  
Employee e = null;  
double clientCosts = -1;  
double employeeSalary = -1;  
Date db = new Date();
```

```
Service s;
```

```
try  
{  
    st = salon.getServiceTypeByID(selectedServiceType);  
    c = salon.getClientByID(selectedClient);  
    e = salon.getEmployeeByID(selectedEmployee);  
    clientCosts = Double.parseDouble(clientCostText.getText());  
    employeeSalary = Double.parseDouble(employeeSalaryText.getText());
```

```
//date  
db = dateChooser.getDate();
```

```
s = new Service(st, db, e, employeeSalary, c, clientCosts, salon);  
}  
catch(NumberFormatException ne)  
{  
    String msg = ne.getMessage();  
    if(msg.indexOf("\n") != -1)
```

```

msg = msg.substring(msg.indexOf("\\"), msg.lastIndexOf("\")+1);
JOptionPane.showMessageDialog(this, "You inputted wrong number: " + msg + ".", "Error",
JOptionPane.ERROR_MESSAGE);
return;
}
catch(InvalidNameException ie)
{
JOptionPane.showMessageDialog(this, ie.getMessage() + "Bab symbol: " +
ie.getInvalidSymbol(), "Error", JOptionPane.ERROR_MESSAGE);
return;
}
catch(Exception ignor)
{
log.error("Problem with adding service: " + ignor);
//ignor.printStackTrace();
return;
}

//REL
s.setRELEVANT(state);

salon.addService(s);

try
{
HibHundler.initFactoryAndSession();
HibHundler.saveObject(s);
HibHundler.closeFactoryAndSession();
}
catch(*Exception */Throwable ex)
{
log.error("Problem with save into DB service: " + s + ": " + ex);
//ex.printStackTrace();
JOptionPane.showMessageDialog(this, "Problem with save into DB");
}

JOptionPane.showMessageDialog(this, "Service (id=" + s.getID() + ") was saved in Data
Base");

salonGUI.refreshTablesOf_Service();

log.info("New service added.");
}
}

```

AddServiceType.java

```
package labgui;
```

```
import java.lang.*;  
import java.util.*;  
import java.awt.*;  
import javax.swing.*;  
import javax.swing.table.*;  
import java.awt.event.*;
```

```
import lab.*;  
import labgui.*;
```

```
import org.apache.log4j.Logger;
```

```
/**
```

```
 * Форма для добавления типа сервиса
```

```
 *
```

```
 */
```

```
class AddServiceType extends JDialog
```

```
{
```

```
private static final Logger log = Logger.getLogger(AddServiceType.class);
```

```
private SalonGUI salonGUI;
```

```
private Salon salon;
```

```
private final int DEFAULT_WIDTH = 400;
```

```
private final int DEFAULT_HEIGHT = 400;
```

```
private JPanel mainPanel;
```

```
private JButton doneButton;
```

```
private JTextField descriptionText;
```

```
private JTextField currentPriceText;
```

```
private JTextField percentageToEmployeeText;
```

```
private boolean state;
```

```
public AddServiceType(JFrame owner)
```

```
{
```

```
super(owner, "Add specialization", false);
```

```
log.info("Creating new AddServiceType frame...");
```

```
mainPanel = new JPanel();
```

```
/*
```

```
description: []
```

```
currentPrice: []
```

```
percentageToEmployee: []
```



```

Relevant: *
[done]
*/
mainPanel.setLayout(new GridLayout(5, 0, 15, 15));
this.add(mainPanel);
updateAdd();

this.setLocationRelativeTo(owner);
pack();

log.info("AddServiceType frame was created.");
}

/**
 * Обновить содержимое формы
 */
*/
public void updateAdd()
{
mainPanel.removeAll();

salonGUI = SalonGUI.getSalonGUI();
salon = salonGUI.getSalon();

//description
JPanel dp = new JPanel();
dp.add(new JLabel("Description: "));
descriptionText = new JTextField("", 20);
dp.add(descriptionText);
mainPanel.add(dp);

//currentPrice
JPanel cpp = new JPanel();
cpp.add(new JLabel("Price: "));
currentPriceText = new JTextField("", 20);
cpp.add(currentPriceText);
mainPanel.add(cpp);

//percentageToEmployee
JPanel ptep = new JPanel();
ptep.add(new JLabel("Percentage: "));
percentageToEmployeeText = new JTextField("", 20);
ptep.add(percentToEmployeeText);
mainPanel.add(ptep);

//Relevant
JPanel ap = new JPanel();

```

```

ap.add(new JLabel("State: "));
JCheckBox cb = new JCheckBox("relevant");
state = true;
cb.setSelected(state);
cb.addItemListener(new ItemListener() {
    public void itemStateChanged(ItemEvent ie) {
        state = cb.isSelected();
    }
});
ap.add(cb);
mainPanel.add(ap);

//Button
doneButton = new JButton("Done!");
doneButton.addActionListener(aeDone -> {addServiceType();});
mainPanel.add(doneButton);

mainPanel.updateUI();
}

```

```

public Dimension addPreferredSize()
{
    return new Dimension(DEFAULT_WIDTH, DEFAULT_HEIGHT);
}

```

```

@SuppressWarnings( "deprecation" )
private void addServiceType()
{
    log.info("Adding new service type...");
}

```

```

String description = "";
double currentPrice = -1;
float percentageToEmployee = -1;

```

```

ServiceType st;

```

```

try
{
    //description
    description = Salon.checkName4Salon(descriptionText.getText());

    //currentPrice
    currentPrice = Double.parseDouble( currentPriceText.getText() );

    //percentageToEmployee
    percentageToEmployee = Float.parseFloat( percentageToEmployeeText.getText() );
    st = new ServiceType(description, currentPrice, percentageToEmployee, salon);
}

```

```

    }
    catch(NumberFormatException ne)
    {
        String msg = ne.getMessage();
        if(msg.indexOf("\"") != -1)
            msg = msg.substring(msg.indexOf("\""), msg.lastIndexOf("\"")+1);
        JOptionPane.showMessageDialog(this, "You inputted wrong number: " + msg + ".", "Error",
        JOptionPane.ERROR_MESSAGE);
        return;
    }
    catch(InvalidNameException ie)
    {
        JOptionPane.showMessageDialog(this, ie.getMessage() + "Bab symbol: " +
        ie.getInvalidSymbol(), "Error", JOptionPane.ERROR_MESSAGE);
        return;
    }
    catch(Exception ignor)
    {
        log.error("Problem with adding service type: " + ignor);
        //ignor.printStackTrace();
        return;
    }

    //Relevant
    st.relevant(state);
    salon.addServiceType(st);

    try
    {
        HibHundler.initFactoryAndSession();
        HibHundler.saveObject(st);
        HibHundler.closeFactoryAndSession();
    }
    catch(*Exception */Throwable ex)
    {
        log.error("Problem with save into DB service type: " + st + ": " + ex);
        //ex.printStackTrace();
        JOptionPane.showMessageDialog(this, "Problem with save into DB");
    }

    JOptionPane.showMessageDialog(this, "Service type (id=" + st.getID() + ") was saved in Data
    Base");

    salonGUI.refreshTablesOf_ServiceTypes();

    log.info("New service type added.");
}

```

```
}
```

EditClient.java

```
package labgui;

import java.lang.*;
import java.util.*;
import java.awt.*;
import javax.swing.*;
import javax.swing.table.*;
import java.awt.event.*;

import lab.*;
import labgui.*;

import org.apache.log4j.Logger;

/**
 * Форма для редактирования клиента
 */
class EditClient extends JDialog
{
    private static final Logger log = Logger.getLogger(EditClient.class);

    private SalonGUI salonGUI;
    private Salon salon;

    private final int DEFAULT_WIDTH = 500;
    private final int DEFAULT_HEIGHT = 750;

    private JPanel mainPanel;
    private JButton doneButton;

    private JTextField firstNameText;
    private JTextField secondNameText;
    private DateChooser dateChooser;
    private JTextField passportText;
    private boolean gender;
    private boolean state;
    private JTextField priorityText;

    public EditClient(JFrame owner, Client c)
    {
```

```

super(owner, "Edit client", false);

log.info("Creating new EditClient frame...");

mainPanel = new JPanel();
/*
id
firstName: []
secondName: []
birthday: []
passport: []
gender: * *
BANNED: *
priority: []
[done]
*/
mainPanel.setLayout(new GridLayout(9, 0, 15, 15));
this.add(mainPanel);
updateEdit(c);

this.setLocationRelativeTo(owner);
pack();

log.info("EditClient frame was created.");
}

/**
 * Обновить содержимое формы
 *
 * @param c клиент, для которого обновляется
 */
public void updateEdit(Client c)
{
mainPanel.removeAll();

salonGUI = SalonGUI.getSalonGUI();
salon = salonGUI.getSalon();

//id
JPanel ip = new JPanel();
ip.add(new JLabel("ID is " + c.getID()));
mainPanel.add(ip);

//firstName
JPanel fnp = new JPanel();
fnp.add(new JLabel("First name: "));
firstNameText = new JTextField(c.getFirstName(), 20);

```

```

fnp.add(firstNameText);
mainPanel.add(fnp);

//secondName
JPanel cnp = new JPanel();
cnp.add(new JLabel("Second name: "));
secondNameText = new JTextField(c.getSecondName(), 20);
cnp.add(secondNameText);
mainPanel.add(cnp);

//birthday
JPanel bp = new JPanel();
bp.add(new JLabel("Birthday: "));
dateChooser = new DateChooser();
dateChooser.setDate(c.getBirthday());
bp.add(dateChooser);
mainPanel.add(bp);

//passport
JPanel pp = new JPanel();
pp.add(new JLabel("Passport: "));
passportText = new JTextField(c.getPassport(), 20);
pp.add(passportText);
mainPanel.add(pp);

//gender
JPanel gp = new JPanel();

gender = c.getGender();
JPanel gender4Ratio = new JPanel();
JRadioButton genderMaleRatio = new JRadioButton("Male");
JRadioButton genderFemaleRatio = new JRadioButton("Female");
ActionListener genderActionListener = ae ->
{
    if(genderMaleRatio.isSelected())
        gender = true;
    else if(genderFemaleRatio.isSelected())
        gender = false;
};
genderMaleRatio.addActionListener(genderActionListener);
genderFemaleRatio.addActionListener(genderActionListener);
ButtonGroup genderRatioGroup = new ButtonGroup();
genderRatioGroup.add(genderMaleRatio);
genderRatioGroup.add(genderFemaleRatio);
gender4Ratio.add(genderMaleRatio);
gender4Ratio.add(genderFemaleRatio);
if(gender == true)

```

```

genderMaleRatio.setSelected(true);
else
genderFemaleRatio.setSelected(true);

gp.add(new JLabel("Gender: "));
gp.add(gender4Ratio);
mainPanel.add(gp);

//BANNED
JPanel ap = new JPanel();
ap.add(new JLabel("Banned: "));
JCheckBox cb = new JCheckBox("is banned?");
state = c.isBanned();
cb.setSelected(state);
cb.addItemListener(new ItemListener() {
    public void itemStateChanged(ItemEvent ie) {
        state = cb.isSelected();
    }
});
ap.add(cb);
mainPanel.add(ap);

//priority
JPanel prp = new JPanel();
prp.add(new JLabel("Priority: "));
priorityText = new JTextField("" + c.getPriority(), 20);
prp.add(priorityText);
mainPanel.add(prp);

//Button
doneButton = new JButton("Done!");
doneButton.addActionListener(aeDone -> {changeClient(c);});
mainPanel.add(doneButton);

mainPanel.updateUI();
}

public Dimension getPreferredSize()
{
    return new Dimension(DEFAULT_WIDTH, DEFAULT_HEIGHT);
}

@SuppressWarnings( "deprecation" )
private void changeClient(Client c)
{
    try
    {

```

```

log.info("Trying to edit client...");

//names
c.setName(Salon.checkName4Salon(firstNameText.getText()),
Salon.checkName4Salon(secondNameText.getText()));

//birthday
Date d = dateChooser.getDate();
c.setBirthday(d);

//passport
c.setPassport(Salon.checkName4Salon(passportText.getText()));

//gender
c.setGender(gender);

//BANNED
c.ban(state);

//priority
c.setPriority( Integer.parseInt( priorityText.getText() ) );
}
catch(NumberFormatException ne)
{
String msg = ne.getMessage();
if(msg.indexOf("\\"") != -1)
msg = msg.substring(msg.indexOf("\\""), msg.lastIndexOf("\\"")+1);
JOptionPane.showMessageDialog(this, "You inputted wrong number: " + msg + ".", "Error",
JOptionPane.ERROR_MESSAGE);
return;
}
catch(InvalidNameException ie)
{
JOptionPane.showMessageDialog(this, ie.getMessage() + "Bab symbol: " +
ie.getInvalidSymbol(), "Error", JOptionPane.ERROR_MESSAGE);
return;
}
catch(Exception ignor)
{
log.error("Problem with editing client: " + c + ": " + ignor);
//ignor.printStackTrace();
return;
}
salonGUI.refreshTablesOf_Clients();

log.info("Client was edited.");
}

```



```
}
```

EditEmployee.java

```
package labgui;

import java.lang.*;
import java.util.*;
import java.awt.*;
import javax.swing.*;
import javax.swing.table.*;
import java.awt.event.*;

import lab.*;
import labgui.*;

import org.apache.log4j.Logger;

/**
 * Форма для редактирования сотрудника
 */
class EditEmployee extends JDialog
{
    private static final Logger log = Logger.getLogger(EditEmployee.class);

    private SalonGUI salonGUI;
    private Salon salon;

    private final int DEFAULT_WIDTH = 500;
    private final int DEFAULT_HEIGHT = 750;

    private JPanel mainPanel;
    private JButton doneButton;

    private JTextField firstNameText;
    private JTextField secondNameText;
    private DateChooser dateChooser;
    private JTextField passportText;
    private boolean gender;
    private boolean state;
    private JTextField permittedServicesText;

    public EditEmployee(JFrame owner, Employee e)
    {
        super(owner, "Edit employee", false);
    }
}
```

```

log.info("Creating new EditEmployee frame...");

mainPanel = new JPanel();
/*
id
firstName: []
secondName: []
birthday: []
passport: []
gender: * *
ACTIVE: *
permittedServices: []
[done]
*/
mainPanel.setLayout(new GridLayout(9, 0, 15, 15));
this.add(mainPanel);
updateEdit(e);

this.setLocationRelativeTo(owner);
pack();

log.info("EditEmployee frame was created.");
}

/**
 * Обновить содержимое формы
 *
 * @param e сотрудник, для которого обновляется
 */
public void updateEdit(Employee e)
{
mainPanel.removeAll();

salonGUI = SalonGUI.getSalonGUI();
salon = salonGUI.getSalon();

//id
JPanel ip = new JPanel();
ip.add(new JLabel("ID is " + e.getID()));
mainPanel.add(ip);

//firstName
JPanel fnp = new JPanel();
fnp.add(new JLabel("First name: "));
firstNameText = new JTextField(e.getFirstName(), 20);
fnp.add(firstNameText);

```

```
mainPanel.add(fnp);
```

```
//secondName
```

```
JPanel cnp = new JPanel();
```

```
cnp.add(new JLabel("Second name: "));
```

```
secondNameText = new TextField(e.getSecondName(), 20);
```

```
cnp.add(secondNameText);
```

```
mainPanel.add(cnp);
```

```
//birthday
```

```
JPanel bp = new JPanel();
```

```
bp.add(new JLabel("Birthday: "));
```

```
dateChooser = new DateChooser();
```

```
dateChooser.setDate(e.getBirthday());
```

```
bp.add(dateChooser);
```

```
mainPanel.add(bp);
```

```
//passport
```

```
JPanel pp = new JPanel();
```

```
pp.add(new JLabel("Passport: "));
```

```
passportText = new TextField(e.getPassport(), 20);
```

```
pp.add(passportText);
```

```
mainPanel.add(pp);
```

```
//gender
```

```
JPanel gp = new JPanel();
```

```
gender = e.getGender();
```

```
JPanel gender4Ratio = new JPanel();
```

```
JRadioButton genderMaleRatio = new JRadioButton("Male");
```

```
JRadioButton genderFemaleRatio = new JRadioButton("Female");
```

```
ActionListener genderActionListener = ae ->
```

```
{
```

```
if(genderMaleRatio.isSelected())
```

```
gender = true;
```

```
else if(genderFemaleRatio.isSelected())
```

```
gender = false;
```

```
};
```

```
genderMaleRatio.addActionListener(genderActionListener);
```

```
genderFemaleRatio.addActionListener(genderActionListener);
```

```
ButtonGroup genderRatioGroup = new ButtonGroup();
```

```
genderRatioGroup.add(genderMaleRatio);
```

```
genderRatioGroup.add(genderFemaleRatio);
```

```
gender4Ratio.add(genderMaleRatio);
```

```
gender4Ratio.add(genderFemaleRatio);
```

```
if(gender == true)
```

```
genderMaleRatio.setSelected(true);
```

```

else
genderFemaleRatio.setSelected(true);

gp.add(new JLabel("Gender: "));
gp.add(gender4Ratio);
mainPanel.add(gp);

//ACTIVE
JPanel ap = new JPanel();
ap.add(new JLabel("State: "));
JCheckBox cb = new JCheckBox("relevant");
state = e.isActive();
cb.setSelected(state);
cb.addItemListener(new ItemListener() {
public void itemStateChanged(ItemEvent ie) {
state = cb.isSelected();
}
});
ap.add(cb);
mainPanel.add(ap);

//permittedServices
JPanel psp = new JPanel();
psp.add(new JLabel("Permitted services: "));
Set<ServiceType> mst = e.getMasteredServices();
String buffS = "";
for(ServiceType st : mst)
buffS += st.getID() + " ";
permittedServicesText = new JTextField(buffS, 15);
psp.add(permittedServicesText);
mainPanel.add(psp);

//Button
doneButton = new JButton("Done!");
doneButton.addActionListener(aeDone -> {changeEmployee(e);});
mainPanel.add(doneButton);

mainPanel.updateUI();
}

public Dimension getPreferredSize()
{
return new Dimension(DEFAULT_WIDTH, DEFAULT_HEIGHT);
}

@SuppressWarnings( "deprecation" )
private void changeEmployee(Employee e)

```

```

{
try
{
log.info("Trying to edit employee...");

//names
e.setName(Salon.checkName4Salon(firstNameText.getText()),
Salon.checkName4Salon(secondNameText.getText()));

//birthday
Date d = dateChooser.getDate();
e.setBirthday(d);

//passport
e.setPassport(Salon.checkName4Salon(passportText.getText()));

//gender
e.setGender(gender);

//ACTIVE
e.active(state);

//permittedServices
String[] buffSs;
if(!permittedServicesText.getText().trim().equals(""))
{
buffSs = permittedServicesText.getText().trim().split(" ");
int[] pss = new int[buffSs.length];
for(int i = 0; i < pss.length; ++i)
pss[i] = Integer.parseInt(buffSs[i]);
e.forbidAllServices();
for(int i = 0; i < pss.length; ++i)
{
ServiceType buffST = salon.getServiceTypeByID(pss[i]);
if(buffST == null)
throw new NumberFormatException("" + pss[i]);
e.addService(buffST);
}
}
else
e.forbidAllServices();
}
catch(NumberFormatException ne)
{
String msg = ne.getMessage();
if(msg.indexOf("\\" != -1)

```

```

msg = msg.substring(msg.indexOf("\\"), msg.lastIndexOf("\")+1);
JOptionPane.showMessageDialog(this, "You inputted wrong number: " + msg + ".", "Error",
JOptionPane.ERROR_MESSAGE);
return;
}
catch(InvalidNameException ie)
{
JOptionPane.showMessageDialog(this, ie.getMessage() + "Bab symbol: " +
ie.getInvalidSymbol(), "Error", JOptionPane.ERROR_MESSAGE);
return;
}
catch(Exception ignor)
{
log.error("Problem with editing employee: " + e + ": " + ignor);
//ignor.printStackTrace();
return;
}
salonGUI.refreshTablesOf_Employees();

log.info("Employee was edited.");
}
}

```

EditService.java

```

package labgui;

import java.lang.*;
import java.util.*;
import java.awt.*;
import javax.swing.*;
import javax.swing.table.*;
import java.awt.event.*;

import lab.*;
import labgui.*;

import org.apache.log4j.Logger;

/**
 * Форма для редактирования сервиса
 */
class EditService extends JDialog
{
private static final Logger log = Logger.getLogger(EditService.class);

```

```

private SalonGUI salonGUI;
private Salon salon;

private final int DEFAULT_WIDTH = 200;
private final int DEFAULT_HEIGHT = 200;

private JPanel mainPanel;
private JButton doneButton;

private boolean state;

public EditService(JFrame owner, Service s)
{
    super(owner, "Edit specialization", false);

    log.info("Creating new EditService frame...");

    mainPanel = new JPanel();
    /*
    id
    REL: *
    [done]
    */
    mainPanel.setLayout(new GridLayout(3, 0, 15, 15));
    this.add(mainPanel);
    updateEdit(s);

    this.setLocationRelativeTo(owner);
    pack();

    log.info("EditService frame was created.");
}

/**
 * Обновить содержимое формы
 *
 * @param s сервис, для которого обновляется
 */
public void updateEdit(Service s)
{
    mainPanel.removeAll();

    salonGUI = SalonGUI.getSalonGUI();
    salon = salonGUI.getSalon();

    //id

```

```

JPanel ip = new JPanel();
ip.add(new JLabel("ID is " + s.getID()));
mainPanel.add(ip);

//REL
JPanel rp = new JPanel();
rp.add(new JLabel("State: "));
JCheckBox cb = new JCheckBox("relevant");
state = s.isRELEVANT();
cb.setSelected(state);
cb.addItemListener(new ItemListener() {
    public void itemStateChanged(ItemEvent ie) {
        state = cb.isSelected();
    }
});
rp.add(cb);
mainPanel.add(rp);

//Button
doneButton = new JButton("Done!");
doneButton.addActionListener(aeDone -> {changeService(s);});
mainPanel.add(doneButton);

mainPanel.updateUI();
}

public Dimension getPreferredSize()
{
    return new Dimension(DEFAULT_WIDTH, DEFAULT_HEIGHT);
}

@SuppressWarnings( "deprecation" )
private void changeService(Service s)
{
    log.info("Trying to edit service...");

    //REL
    s.setRELEVANT(state);
    salonGUI.refreshTablesOf_Service();

    log.info("Service was edited.");
}
}

```

EditServiceType.java


```
package labgui;
```

```
import java.lang.*;  
import java.util.*;  
import java.awt.*;  
import javax.swing.*;  
import javax.swing.table.*;  
import java.awt.event.*;
```

```
import lab.*;  
import labgui.*;
```

```
import org.apache.log4j.Logger;
```

```
/**
```

```
 * Форма для редактирования типа сервиса
```

```
 *
```

```
 */
```

```
class EditServiceType extends JDialog
```

```
{
```

```
private static final Logger log = Logger.getLogger(EditServiceType.class);
```

```
private SalonGUI salonGUI;
```

```
private Salon salon;
```

```
private final int DEFAULT_WIDTH = 400;
```

```
private final int DEFAULT_HEIGHT = 400;
```

```
private JPanel mainPanel;
```

```
private JButton doneButton;
```

```
private JTextField descriptionText;
```

```
private JTextField currentPriceText;
```

```
private JTextField percentageToEmployeeText;
```

```
private boolean state;
```

```
public EditServiceType(JFrame owner, ServiceType st)
```

```
{
```

```
super(owner, "Edit specialization", false);
```

```
log.info("Creating new EditServiceType frame...");
```

```
mainPanel = new JPanel();
```

```
/*
```

```
id
```

```
description: []
```

```
currentPrice: []
```

```

percentageToEmployee: []
Relevant: *
[done]
*/
mainPanel.setLayout(new GridLayout(6, 0, 15, 15));
this.add(mainPanel);
updateEdit(st);

this.setLocationRelativeTo(owner);
pack();

log.info("EditServiceType frame was created.");
}

/**
 * Обновить содержимое формы
 *
 * @param st тип сервиса, для которого обновляется
 */
public void updateEdit(ServiceType st)
{
    mainPanel.removeAll();

    salonGUI = SalonGUI.getSalonGUI();
    salon = salonGUI.getSalon();

    //id
    JPanel ip = new JPanel();
    ip.add(new JLabel("ID is " + st.getID()));
    mainPanel.add(ip);

    //description
    JPanel dp = new JPanel();
    dp.add(new JLabel("Description: "));
    descriptionText = new JTextField(st.getDescription(), 20);
    dp.add(descriptionText);
    mainPanel.add(dp);

    //currentPrice
    JPanel cpp = new JPanel();
    cpp.add(new JLabel("Price: "));
    currentPriceText = new JTextField("" + st.getCurrentPrice(), 20);
    cpp.add(currentPriceText);
    mainPanel.add(cpp);

    //percentageToEmployee
    JPanel ptep = new JPanel();

```

```

ptep.add(new JLabel("Percentage: "));
percentageToEmployeeText = new JTextField("" + st.getPercent(), 20);
ptep.add(permissionToEmployeeText);
mainPanel.add(ptep);

//Relevant
JPanel ap = new JPanel();
ap.add(new JLabel("State: "));
JCheckBox cb = new JCheckBox("relevant");
state = st.isRelevant();
cb.setSelected(state);
cb.addItemListener(new ItemListener() {
    public void itemStateChanged(ItemEvent ie) {
        state = cb.isSelected();
    }
});
ap.add(cb);
mainPanel.add(ap);

//Button
doneButton = new JButton("Done!");
doneButton.addActionListener(aeDone -> {changeServiceType(st);});
mainPanel.add(doneButton);

mainPanel.updateUI();
}

public Dimension getPreferredSize()
{
    return new Dimension(DEFAULT_WIDTH, DEFAULT_HEIGHT);
}

@SuppressWarnings( "deprecation" )
private void changeServiceType(ServiceType st)
{
    try
    {
        log.info("Trying to edit service type...");
    }

//description
st.changeDescription( Salon.checkName4Salon(descriptionText.getText()) );

//currentPrice
st.setPrice( Double.parseDouble( currentPriceText.getText() ) );

//percentageToEmployee
st.setPercent( Float.parseFloat( percentageToEmployeeText.getText() ) );

```

```

    }
    catch(NumberFormatException ne)
    {
        String msg = ne.getMessage();
        if(msg.indexOf("\\"") != -1)
            msg = msg.substring(msg.indexOf("\\""), msg.lastIndexOf("\\"")+1);
        JOptionPane.showMessageDialog(this, "You inputted wrong number: " + msg + ".", "Error",
        JOptionPane.ERROR_MESSAGE);
        return;
    }
    catch(InvalidNameException ie)
    {
        JOptionPane.showMessageDialog(this, ie.getMessage() + "Bab symbol: " +
        ie.getInvalidSymbol(), "Error", JOptionPane.ERROR_MESSAGE);
        return;
    }
    catch(Exception ignor)
    {
        log.error("Problem with editing service type: " + st + ": " + ignor);
        //ignor.printStackTrace();
        return;
    }
    //Relevant
    st.relevant(state);

    salonGUI.refreshTablesOf_ServiceTypes();

    log.info("Service type was edited.");
}
}

```

DateChooser.java

```

package labgui;

import java.lang.*;
import java.util.*;
import java.awt.*;
import javax.swing.*;
import javax.swing.table.*;
import java.awt.event.*;

/**
 * Панель для выбора даты
 *
 */

```

```

@SuppressWarnings( "deprecation" )
public class DateChooser extends JPanel
{
    private String[] yComboStr;
    private String[] mComboStr;
    private String[] dComboStr;

    private JComboBox yCombo;
    private JComboBox mCombo;
    private JComboBox dCombo;

    private JTextField textDate;

    private int selectedYear;
    private int selectedMonth;
    private int selectedDate;

    public DateChooser()
    {
        int currentY = (new Date()).getYear() + 1900;
        yComboStr = get_yComboStr(currentY-100, currentY+1);
        mComboStr = get_mComboStr();
        dComboStr = get_dComboStr();

        yCombo = new JComboBox<String>(yComboStr);
        mCombo = new JComboBox<String>(mComboStr);
        dCombo = new JComboBox<String>(dComboStr);
        textDate = new JTextField();

        ActionListener yActionListener = new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                JComboBox box = (JComboBox)e.getSource();
                selectedYear = Integer.parseInt((String)box.getSelectedItem());
                textDate.setText("You selected: " + doStringDate(getDate()));
            }
        };
        yCombo.addActionListener(yActionListener);

        ActionListener mActionListener = new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                JComboBox box = (JComboBox)e.getSource();
                selectedMonth = Integer.parseInt((String)box.getSelectedItem());
                textDate.setText("You selected: " + doStringDate(getDate()));
            }
        };
        mCombo.addActionListener(mActionListener);
    }
}

```

```

ActionListener dActionListener = new ActionListener() {
public void actionPerformed(ActionEvent e) {
JComboBox box = (JComboBox)e.getSource();
selectedDate = Integer.parseInt((String)box.getSelectedItem());
textDate.setText("You selected: " + doStringDate(getDate()));
}
};
dCombo.addActionListener(dActionListener);

//yCombo.setSelectedIndex(0);
//mCombo.setSelectedIndex(0);
//dCombo.setSelectedIndex(0);
setDate(new Date());

this.setLayout(new GridLayout(2, 1/*, 15, 15*/));

JPanel downPanel = new JPanel();
downPanel.setLayout(new GridLayout(1, 3/*, 15, 15*/));

JPanel dPanel = new JPanel();
dPanel.add(new JLabel("Day: "));
dPanel.add(dCombo);

JPanel mPanel = new JPanel();
mPanel.add(new JLabel("Month: "));
mPanel.add(mCombo);

JPanel yPanel = new JPanel();
yPanel.add(new JLabel("Year: "));
yPanel.add(yCombo);

downPanel.add(dPanel);
downPanel.add(mPanel);
downPanel.add(yPanel);

textDate.setText("Today is " + doStringDate(new Date()));
textDate.setEditable(false);

this.add(textDate);
this.add(downPanel);

}

/**
 * Получить дату, выбранную на панели
 */

```

```

* @return дата, выбранная на панели
*/
public Date getDate()
{
    Date resDate = new Date();
    resDate.setYear(selectedYear - 1900);
    resDate.setMonth(selectedMonth-1);
    resDate.setDate(selectedDate);
    return resDate;
}

/**
* Установить дату на панели
*
* @param date дата, которую нужно установить на панели
*/
public void setDate(Date date)
{
    int y = date.getYear() + 1900;
    int m = date.getMonth() + 1;
    int day = date.getDate();

    yCombo.setSelectedItem("" + y);
    mCombo.setSelectedItem("" + m);
    dCombo.setSelectedItem("" + day);
}

private String[] get_yComboStr(int b, int e)
{
    int i, j;
    String[] res = new String[(e-b)+1];
    for(i = e, j = 0; i >= b; --i, ++j)
        res[j] = "" + i;
    return res;
}

private String[] get_mComboStr()
{
    int b = 1;
    int e = 12;
    int i, j;
    String[] res = new String[(e-b)+1];
    for(i = b, j = 0; i <= e; ++i, ++j)
        res[j] = "" + i;
    return res;
}

```

```

private String[] get_dComboStr()
{
    int b = 1;
    int e = 31;
    int i, j;
    String[] res = new String[(e-b)+1];
    for(i = b, j = 0; i <= e; ++i, ++j)
        res[j] = "" + i;
    return res;
}

/**
 * Преобразует дату в строку
 *
 * @param d преобразуемая дата
 * @return строка с датой d
 */
public static String doStringDate(Date d)
{
    String toOut = "";
    toOut += (d.getDate() > 9 ? d.getDate() : "0" + d.getDate()) + ".";
    toOut += (d.getMonth() + 1 > 9 ? d.getMonth() + 1 : "0" + (d.getMonth() + 1)) + ".";
    toOut += (d.getYear() + 1900);
    return toOut;
}
}

```

LoadingMessage.java

```

package labgui;

import java.lang.*;
import java.util.*;
import java.awt.*;
import javax.swing.*;
import javax.swing.table.*;
import java.awt.event.*;

/**
 * Форма для загрузки. Во время загрузки чего либо можно вызвать эту форму, чтобы
 * пользователю было известно, что именно сейчас происходит
 *
 */
public class LoadingMessage extends JDialog
{
    private final int DEFAULT_WIDTH = 330;
}

```



```
private final int DEFAULT_HEIGHT = 100;
```

```
private JPanel picPanel;  
private ArrayList<JLabel> imgs;
```

```
private String msg;  
private JLabel msgLabel;
```

```
private boolean ALIVE;
```

```
public LoadingMessage(JFrame owner, String msg)  
{  
    super(owner, "Loading...", true);  
    setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);
```

```
    this.msg = msg;  
    this.ALIVE = false;
```

```
    picPanel = new JPanel();
```

```
    imgs = new ArrayList<JLabel>();  
    imgs.add( new JLabel(new ImageIcon(LoadingMessage.class.getResource("/img/  
loadPic1.png"))));  
    imgs.add( new JLabel(new ImageIcon(LoadingMessage.class.getResource("/img/  
loadPic2.png"))));  
    imgs.add( new JLabel(new ImageIcon(LoadingMessage.class.getResource("/img/  
loadPic3.png"))));  
    imgs.add( new JLabel(new ImageIcon(LoadingMessage.class.getResource("/img/  
loadPic4.png"))));  
    imgs.add( new JLabel(new ImageIcon(LoadingMessage.class.getResource("/img/  
loadPic3.png"))));  
    imgs.add( new JLabel(new ImageIcon(LoadingMessage.class.getResource("/img/  
loadPic4.png"))));  
    imgs.add( new JLabel(new ImageIcon(LoadingMessage.class.getResource("/img/  
loadPic3.png"))));
```

```
    msgLabel = new JLabel(msg);  
    this.add(picPanel);
```

```
    this.setLocationRelativeTo(owner);  
    pack();  
}
```

```
public Dimension getPreferredSize()  
{  
    return new Dimension(DEFAULT_WIDTH, DEFAULT_HEIGHT);  
}
```

```

/**
 * Показать форму загрузки
 *
 * @param SHOWED true=показать, false=скрыть
 */
public void showLoad(boolean SHOWED)
{
    if(SHOWED == true)
    {
        ALIVE = true;
        Runnable task = () ->
        {
            picPanelAnimator();
        };
        new Thread(task).start();
        this.setVisible(true);
    }
    else
    {
        ALIVE = false;
        this.setVisible(false);
    }
}

private void picPanelAnimator()
{
    try
    {
        if(imgs.size() > 0)
        {
            int tick = 0;
            while(ALIVE)
            {
                picPanel.removeAll();

                picPanel.add(msgLabel);

                picPanel.add( imgs.get(tick) );

                tick = (tick+1)%imgs.size();

                picPanel.updateUI();

                Thread.sleep(500);
            }
        }
    }
}

```

```

else
picPanel.add(msgLabel);
}
catch(Exception e)
{
e.printStackTrace();
}
}
}
}

```

SalonGUI.java

```

package labgui;

```

```

import java.net.URL;
import java.lang.*;
import java.util.*;
import java.awt.*;
import javax.swing.*;
import javax.swing.table.*;
import java.awt.event.*;

```

```

import lab.*;
import labgui.*;

```

```

import org.apache.log4j.Logger;

```

```

/**
 * Главное окно приложения
 *
 */

```

```

public class SalonGUI

```

```

{
private static final Logger log = Logger.getLogger(SalonGUI.class);

```

```

private static SalonGUI salonGUI;
private Salon salon;

```

```

private static String[] employeesCol = {"ID", "Full name", "Gender", "Passport", "Birthday",
"Specializations", "Active"};
private static String[] clientsCol = {"ID", "Full name", "Gender", "Passport", "Birthday",
"Priority", "BANNED"};
private static String[] servicesCol = {"ID", "Service type", "Date", "Employee",
"Employee's cut", "Client", "Price", "Relevance"};
private static String[] serviceTypesCol = {"ID", "Description", "Price", "Percent, %",
"Relevance"};

```

```
private boolean DBLOADED;

private JFrame mainFrame;
private JButton saveButton;
private JButton openButton;
private JButton addButton;
private JButton editButton;
//private JButton deleteButton;
private JButton taskButton;

private JPanel bottomPanel;
private JLabel bottomLabel;
private JTextField bottomText;
private JButton bottomButton;
private JCheckBox consideredCheckBox;

private JToolBar soaedToolBar;

private JPanel currentTable;

private DefaultTableModel employeeModel;
private JTable employeeTable;
private JScrollPane employeePane;

private DefaultTableModel clientModel;
private JTable clientTable;
private JScrollPane clientPane;

private DefaultTableModel serviceModel;
private JTable serviceTable;
private JScrollPane servicePane;

private DefaultTableModel serviceTypeModel;
private JTable serviceTypeTable;
private JScrollPane serviceTypePane;

private JMenuBar chooseTableMenuBar;
private JMenu chooseTableMenu;

private EditEmployee employeeEditor;
private EditClient clientEditor;
private EditService serviceEditor;
private EditServiceType serviceTypeEditor;

private AddEmployee employeeAdder;
private AddClient clientAdder;
```

```
private AddService serviceAdder;  
private AddServiceType serviceTypeAdder;
```

```
private TaskHundler taskHundler;
```

```
private JScrollPane curPane;
```

```
private String filter;  
private String[] filters;
```

```
//true = не выводить неактивные, забанненные, неактуальные
```

```
private boolean CONSIDERED;
```

```
private LoadingMessage loadLoading;  
private LoadingMessage saveLoading;
```

```
/**
```

```
 * Запуск GUI
```

```
 *
```

```
 */
```

```
public static void start()  
{  
    salonGUI = new SalonGUI();  
    salonGUI.show();  
}
```

```
private void show()  
{  
    DBLOADED = false;
```

```
log.info("Preparing main frame...");
```

```
mainFrame = new JFrame("Frame of salon");  
mainFrame.setSize(1280, 720);  
mainFrame.setLocation(50, 51);  
mainFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

```
openButton = new JButton(new ImageIcon(SalonGUI.class.getResource("/img/  
openButton.png")));  
openButton.setToolTipText("Open DB");  
openButton.addActionListener(event ->  
{  
    loadSalon();  
    if(salon != null)  
    {  
        PrapareTables();  
        curPane = /*curPane==null?*/employeePane/*:curPane*/;  
        refresh(curPane);
```

```
DBLOADED = true;
}
});
```

```
saveButton = new JButton(new ImageIcon(SalonGUI.class.getResource("/img/
saveButton.png")));
saveButton.setToolTipText("Save into DB");
saveButton.addActionListener(event -> {if(checkPrepare() == true) saveSalon();});
```

```
addButton = new JButton(new ImageIcon(SalonGUI.class.getResource("/img/
addButton.png")));
addButton.setToolTipText("Add row");
addButton.addActionListener(event -> {if(checkPrepare() == true) addItem(curPane);});
```

```
editButton = new JButton(new ImageIcon(SalonGUI.class.getResource("/img/
editButton.png")));
editButton.setToolTipText("Edit row");
editButton.addActionListener(event -> {if(checkPrepare() == true) editItem(curPane);});
```

```
//deleteButton = new JButton(new ImageIcon(SalonGUI.class.getResource("/img/
deleteButton.png")));
//deleteButton.setToolTipText("Delete row");
//deleteButton.addActionListener(event -> {System.out.println("TunA YDAJluJlOCb...");});
```

```
taskButton = new JButton(new ImageIcon(SalonGUI.class.getResource("/img/
taskButton.png")));
taskButton.setToolTipText("Task");
taskButton.addActionListener(event -> {if(checkPrepare() == true) doTask();});
```

```
soaedToolBar = new JToolBar("Tool bar");
soaedToolBar.add(openButton);
soaedToolBar.add(saveButton);
soaedToolBar.add(addButton);
soaedToolBar.add(editButton);
//soaedToolBar.add(deleteButton);
soaedToolBar.add(taskButton);
```

```
mainFrame.setLayout(new BorderLayout());
mainFrame.add(soaedToolBar, BorderLayout.NORTH);
```

```
//PrapareTables();
currentTable = new JPanel();
mainFrame.add(currentTable, BorderLayout.CENTER);
```

```
bottomLabel = new JLabel("Search: ");
bottomText = new JTextField("", 25);
bottomText.addActionListener(eve ->
```

```

{
if(checkPrepare() == true)
{
filter = bottomText.getText();
filters = filter.split(" ");
refreshTables();
}
});
bottomButton = new JButton(new ImageIcon(SalonGUI.class.getResource("/img/
searchButton.png")));
bottomButton.setToolTipText("Find text");
bottomButton.addActionListener(event ->
{
if(checkPrepare() == true)
{
filter = bottomText.getText();
filters = filter.split(" ");
refreshTables();
//System.out.println("TunA uWET...");
}
});
consideredCheckBox = new JCheckBox("Considered");
CONSIDERED = true;
consideredCheckBox.setSelected(CONSIDERED);
consideredCheckBox.addItemListener(new ItemListener() {
public void itemStateChanged(ItemEvent ie) {
if(checkPrepare() == true)
{
CONSIDERED = consideredCheckBox.isSelected();
refreshTables();
}
}
});

bottomPanel = new JPanel();
bottomPanel.add(bottomLabel);
bottomPanel.add(bottomText);
bottomPanel.add(bottomButton);
bottomPanel.add(consideredCheckBox);

mainFrame.add(bottomPanel, BorderLayout.SOUTH);

JMenuItem chooseEmployeesTable = new JMenuItem("Employees");
chooseEmployeesTable.addActionListener(event ->
{
if(checkPrepare() == true)
{curPane = employeePane; refresh(curPane);}

```

```
});
```

```
JMenuItem chooseClientsTable = new JMenuItem("Clients");  
chooseClientsTable.addActionListener(event ->  
{  
if(checkPrepare() == true)  
{curPane = clientPane; refresh(curPane);}  
});
```

```
JMenuItem chooseServicesTable = new JMenuItem("Services");  
chooseServicesTable.addActionListener(event ->  
{  
if(checkPrepare() == true)  
{curPane = servicePane; refresh(curPane);}  
});
```

```
JMenuItem chooseServiceTypesTable = new JMenuItem("Service types");  
chooseServiceTypesTable.addActionListener(event ->  
{  
if(checkPrepare() == true)  
{curPane = serviceTypePane; refresh(curPane);}  
});
```

```
chooseTableMenu = new JMenu("Choose table");  
chooseTableMenu.add(chooseEmployeesTable);  
chooseTableMenu.add(chooseClientsTable);  
chooseTableMenu.add(chooseServicesTable);  
chooseTableMenu.add(chooseServiceTypesTable);
```

```
chooseTableMenuBar = new JMenuBar();  
chooseTableMenuBar.add(chooseTableMenu);
```

```
mainFrame.setJMenuBar(chooseTableMenuBar);
```

```
loadLoading = new LoadingMessage(mainFrame, "Please wait while loading datas from  
DB...");  
saveLoading = new LoadingMessage(mainFrame, "Please wait while saving datas in DB...");
```

```
mainFrame.setIconImage(new ImageIcon(SalonGUI.class.getResource("/img/  
icon.png")).getImage());
```

```
log.info("Main frame was formed.");
```

```
mainFrame.setVisible(true);  
}
```

```
private void PrapareTables()
```



```

{
log.info("Preparing tables...");

initTablesOf_Employees();

initTablesOf_Clients();

initTablesOf_Services();

initTablesOf_ServiceTypes();
//currentTable = new JPanel();
currentTable.setLayout(new BorderLayout());
currentTable.add(employeePane);

log.info("Table prepared.");
}

private boolean checkPrepare()
{
if(DBLOADED == true)
{
return true;
}
else
{
OptionPane.showMessageDialog(mainFrame, "DB is not loaded. Please \"open\" database first.", "Error", JOptionPane.ERROR_MESSAGE);
return false;
}
}

//
=====Employees=====
=====

private void initTablesOf_Employees()
{
employeeModel = getEmployees_DefaultTableModel();

employeeTable = new JTable(employeeModel);
employeeTable.setAutoCreateRowSorter(true);
employeeTable.setEnabled(true);
DefaultTableCellRenderer r = (DefaultTableCellRenderer)
employeeTable.getDefaultRenderer(Integer.class);
r.setHorizontalAlignment(JLabel.LEFT);
employeeTable.getTableHeader().setReorderingAllowed(false);
employeePane = new JScrollPane(employeeTable);
}

```

```

/**
 * Обновляет таблицу сотрудников
 *
 */
public void refreshTablesOf_Employees()
{
    employeeModel = getEmployees_DefaultTableModel();

    employeeTable.setModel(employeeModel);

    refresh(curPane);
}

private DefaultTableModel getEmployees_DefaultTableModel()
{
    Object[][] employeesData = employees2Table(salon.getEmployees());
    DefaultTableModel employeeModel = new DefaultTableModel(employeesData,
    employeesCol)
    {
        @Override
        public boolean isCellEditable(int i, int i1) {
            return false;
        }

        @Override
        public Class getColumnClass(int column)
        {
            if(column == 0)
                return Integer.class;
            else
                return String.class;
        }
    };
    return employeeModel;
}

//
=====Clients=====
=====

private void initTablesOf_Clients()
{
    clientModel = getClients_DefaultTableModel();

    clientTable = new JTable(clientModel);
    clientTable.setAutoCreateRowSorter(true);
}

```

```

clientTable.setEnabled(true);
DefaultTableCellRenderer r = (DefaultTableCellRenderer)
clientTable.getDefaultRenderer(Integer.class);
r.setHorizontalAlignment(JLabel.LEFT);
clientTable.getTableHeader().setReorderingAllowed(false);
clientPane = new JScrollPane(clientTable);
}

/**
 * Обновляет таблицу клиентов
 */
public void refreshTablesOf_Clients()
{
clientModel = getClients_DefaultTableModel();

clientTable.setModel(clientModel);

refresh(curPane);
}

private DefaultTableModel getClients_DefaultTableModel()
{
Object[][] clientsData = clients2Table(salon.getClients());
DefaultTableModel clientModel = new DefaultTableModel(clientsData, clientsCol)
{
@Override
public boolean isCellEditable(int i, int i1) {
return false;
}

@Override
public Class getColumnClass(int column)
{
if(column == 0 || column == 5)
return Integer.class;
else
return String.class;
}
};
return clientModel;
}

//
=====Services=====
=====

```

```

private void initTablesOf_Services()
{
    serviceModel = getServices_DefaultTableModel();
    serviceTable = new JTable(serviceModel);
    serviceTable.setAutoCreateRowSorter(true);
    serviceTable.setEnabled(true);
    DefaultTableCellRenderer r = (DefaultTableCellRenderer)
    serviceTable.getDefaultRenderer(Integer.class);
    r.setHorizontalAlignment(JLabel.LEFT);
    serviceTable.getTableHeader().setReorderingAllowed(false);
    servicePane = new JScrollPane(serviceTable);
}

/**
 * Обновляет таблицу сервисов
 */
public void refreshTablesOf_Service()
{
    serviceModel = getServices_DefaultTableModel();

    serviceTable.setModel(serviceModel);

    refresh(curPane);
}

private DefaultTableModel getServices_DefaultTableModel()
{
    Object[][] servicesData = services2Table(salon.getDeals());
    DefaultTableModel serviceModel = new DefaultTableModel(servicesData, servicesCol)
    {
        @Override
        public boolean isCellEditable(int i, int i1) {
            return false;
        }

        @Override
        public Class getColumnClass(int column)
        {
            if(column == 0)
                return Integer.class;
            else if(column == 4 || column == 6)
                return Double.class;
            else
                return String.class;
        }
    };
};

```

```
return serviceModel;  
}
```

```
//
```

```
=====ServiceTypes=====
```

```
private void initTablesOf_ServiceTypes()
```

```
{  
serviceTypeModel = getServiceTypes_DefaultTableModel();
```

```
serviceTypeTable = new JTable(serviceTypeModel);  
serviceTypeTable.setAutoCreateRowSorter(true);  
serviceTypeTable.setEnabled(true);
```

```
DefaultTableCellRenderer r = (DefaultTableCellRenderer)  
serviceTypeTable.getDefaultRenderer(Integer.class);  
r.setHorizontalAlignment(JLabel.LEFT);  
serviceTypeTable.getTableHeader().setReorderingAllowed(false);  
serviceTypePane = new JScrollPane(serviceTypeTable);  
}
```

```
/**
```

```
* Обновляет таблицу типов сервисов
```

```
*
```

```
*/
```

```
public void refreshTablesOf_ServiceTypes()
```

```
{  
serviceTypeModel = getServiceTypes_DefaultTableModel();
```

```
serviceTypeTable.setModel(serviceTypeModel);  
refresh(curPane);  
}
```

```
private DefaultTableModel getServiceTypes_DefaultTableModel()
```

```
{  
Object[][] serviceTypesData = serviceTypes2Table(salon.getProvidedServices());  
DefaultTableModel serviceTypeModel = new DefaultTableModel(serviceTypesData,  
serviceTypesCol)
```

```
{
```

```
@Override
```

```
public boolean isCellEditable(int i, int i1) {  
return false;  
}
```

```
@Override
```

```
public Class getColumnClass(int column)  
{
```

```

if(column == 0)
return Integer.class;
else if(column == 2)
return Double.class;
else if(column == 3)
return Float.class;
else
return String.class;
}
};
return serviceTypeModel;
}

```

```
//
```

```
=====
=====
```

```

private void refresh(JScrollPane cur)
{
currentTable.removeAll();
currentTable.add(cur);
currentTable.updateUI();
}

```

```

private void refreshTables()
{
if(curPane == employeePane)
{
refreshTablesOf_Employees();
}
else if(curPane == clientPane)
{
refreshTablesOf_Clients();
}
else if(curPane == servicePane)
{
refreshTablesOf_Service();
}
else if(curPane == serviceTypePane)
{
refreshTablesOf_ServiceTypes();
}
}

```

```

private void loadSalon()
{
try

```

```

{
log.info("Trying to load salon from DB...");

removeVisibleWindows();

Runnable task = () ->
{
HibHundler.initFactoryAndSession();
salon = HibHundler.loadSalon();
//String toPrint = salon.toString();
//System.out.println(toPrint);
salon.initAllLazyRecords();
HibHundler.closeFactoryAndSession();
loadLoading.showLoad(false);
};
new Thread(task).start();

loadLoading.showLoad(true);

log.info("Loaded.");
}
catch(Exception e) Throwable e)
{
log.error("Problem with load from DB: ", e);
loadLoading.showLoad(false);
//e.printStackTrace();
JOptionPane.showMessageDialog(mainFrame, "Problem with load from DB", "Error",
JOptionPane.ERROR_MESSAGE);
}

/*Date fD = new Date(); fD.setYear(2005-1900); fD.setMonth(11); fD.setDate(1);
Date eD = new Date(); eD.setYear(2005-1900); eD.setMonth(11); eD.setDate(31);
for(int li = 0; li < 100; ++li) salon.genNewService(fD , eD);*/
}

private void loadSalon(int ignor)
{
log.info("Loading salon from GreareGenerator...");

HibHundler.initFactoryAndSession();
salon = new Salon("Salon Harry Dubua face");
GreatGenerator.makeSalon(salon);
HibHundler.saveSalon(salon);
salon.initAllLazyRecords();
//String toPrint = salon.toString();
//System.out.println(toPrint);

```

```
HibHundler.closeFactoryAndSession();
```

```
log.info("Salon from GreareGenerator was loaded.");  
}
```

```
private void saveSalon()
```

```
{  
try  
{  
log.info("Trying to save salon in DB...");
```

```
Runnable task = () ->
```

```
{  
HibHundler.initFactoryAndSession();  
HibHundler.saveSalon(salon);  
HibHundler.closeFactoryAndSession();  
saveLoading.showLoad(false);  
};  
new Thread(task).start();
```

```
saveLoading.showLoad(true);
```

```
log.info("Salon saved in DB.");
```

```
}  
catch(/*Exception */Throwable e)  
{  
log.error("Problem with save into DB: ", e);  
loadLoading.showLoad(false);  
//e.printStackTrace();  
JOptionPane.showMessageDialog(mainFrame, "Problem with save into DB", "Error",  
JOptionPane.ERROR_MESSAGE);  
}  
}
```

```
private void removeVisibleWindows()
```

```
{  
if(employeeEditor != null)  
employeeEditor.setVisible(false);  
if(clientEditor != null)  
clientEditor.setVisible(false);  
if(serviceEditor != null)  
serviceEditor.setVisible(false);  
if(serviceTypeEditor != null)  
serviceTypeEditor.setVisible(false);  
if(employeeAdder != null)  
employeeAdder.setVisible(false);  
if(clientAdder != null)
```



```

clientAdder.setVisible(false);
if(serviceAdder != null)
serviceAdder.setVisible(false);
if(serviceTypeAdder != null)
serviceTypeAdder.setVisible(false);
}

```

```

private Object[][] employees2Table(Set<Employee> emps)

```

```

{
int i;
int num = 7;
Object[][] res = new Object[emps.size()][7];
for(i = 0; i < res.length; ++i)
res[i] = new Object[num];
i = 0;
for(Employee e : emps)
{
if(filtering(e) == true)
{
res[i][0] = Integer.valueOf(e.getID());
res[i][1] = (e.getName());
res[i][2] = (e.getGender() == true ? "M" : "F");
res[i][3] = (e.getPassport());
res[i][4] = (Salon.doOnlyDate(e.getBirthDay()));

```

```

Set<ServiceType> mst = e.getMasteredServices();
String buffS = "" + mst.size() + ": ";
for(ServiceType st : mst)
buffS += st.getID() + " ";
res[i][5] = (buffS);

```

```

res[i][6] = ("" + e.isActive());
}
else
res[i] = null;
++i;
}
return removeNull(res, num);
}

```

```

private Object[][] clients2Table(Set<Client> clients)

```

```

{
int i;
int num = 7;
Object[][] res = new Object[clients.size()][7];
for(i = 0; i < res.length; ++i)
res[i] = new Object[num];

```

```

i = 0;
for(Client c : clients)
{
if(filtering(c) == true)
{
res[i][0] = Integer.valueOf(c.getID());
res[i][1] = c.getName();
res[i][2] = c.getGender() == true ? "M" : "F";
res[i][3] = c.getPassport();
res[i][4] = Salon.doOnlyDate(c.getBirthDay());

res[i][5] = Integer.valueOf(c.getPriority());

res[i][6] = "" + c.isBanned();
}
else
res[i] = null;
++i;
}
return removeNull(res, num);
}

```

```

private Object[][] services2Table(Set<Service> ss)
{
int i;
int num = 8;
Object[][] res = new Object[ss.size()][];
for(i = 0; i < res.length; ++i)
res[i] = new Object[num];
i = 0;
for(Service s : ss)
{
if(filtering(s) == true)
{
res[i][0] = Integer.valueOf(s.getID());
res[i][1] = s.getServiceType().getDescription() + " - " + s.getServiceType().getID();
res[i][2] = Salon.doOnlyDate(s.getDateBegin());
res[i][3] = s.getEmployee().getName() + " - " + s.getEmployee().getID();
res[i][4] = Double.valueOf(s.getCashReward());
res[i][5] = s.getClient().getName() + " - " + s.getClient().getID();
res[i][6] = Double.valueOf(s.getPrice());
res[i][7] = "" + s.isRELEVANT();
}
else
res[i] = null;
++i;
}
}

```

```
return removeNull(res, num);  
}
```

```
private Object[][] serviceTypes2Table(Set<ServiceType> sts)  
{  
    int i;  
    int num = 5;  
    Object[][] res = new Object[sts.size()][5];  
    for(i = 0; i < res.length; ++i)  
        res[i] = new Object[num];  
    i = 0;  
    for(ServiceType st : sts)  
    {  
        if(filtering(st) == true)  
        {  
            res[i][0] = Integer.valueOf(st.getID());  
            res[i][1] = st.getDescription();  
            res[i][2] = Double.valueOf(st.getCurrentPrice());  
            res[i][3] = Float.valueOf(st.getPercent());  
  
            res[i][4] = "" + st.isRelevant();  
        }  
        else  
            res[i] = null;  
        ++i;  
    }  
    return removeNull(res, num);  
}
```

```
/**  
 * Получить салон  
 *  
 * @return салон  
 */  
public Salon getSalon()  
{  
    return salon;  
}
```

```
/**  
 * Получить salonGUI  
 *  
 * @return salonGUI  
 */  
public static SalonGUI getSalonGUI()  
{  
    return salonGUI;  
}
```

```
}
```

```
//true = показывать
```

```
private boolean filtering(Object obj)
{
    boolean FIRSTCATCH;
    String prop;
    if(obj.getClass() == Employee.class)
    {
        Employee e = (Employee)obj;
        if(CONSIDERED == false)
            FIRSTCATCH = true;
        else if(e.isActive() == false)
            FIRSTCATCH = false;
        else
            FIRSTCATCH = true;
        if(FIRSTCATCH == false)
            return false;
        else
        {
            prop = makeEntityStringProperties(e);
            return checkFilterAndProperties(prop, filters);
        }
    }
    else if(obj.getClass() == Client.class)
    {
        Client c = (Client)obj;
        if(CONSIDERED == false)
            FIRSTCATCH = true;
        else if(c.isBanned() == true)
            FIRSTCATCH = false;
        else
            FIRSTCATCH = true;
        if(FIRSTCATCH == false)
            return false;
        else
        {
            prop = makeEntityStringProperties(c);
            return checkFilterAndProperties(prop, filters);
        }
    }
    else if(obj.getClass() == Service.class)
    {
        Service s = (Service)obj;
        if(CONSIDERED == false)
            FIRSTCATCH = true;
        else if(s.isRELEVANT() == false)
```

```

FIRSTCATCH = false;
else
FIRSTCATCH = true;
if(FIRSTCATCH == false)
return false;
else
{
prop = makeEntityStringProperties(s);
return checkFilterAndProperties(prop, filters);
}
}
else if(obj.getClass() == ServiceType.class)
{
ServiceType st = (ServiceType)obj;
if(CONSIDERED == false)
FIRSTCATCH = true;
else if(st.isRelevant() == false)
FIRSTCATCH = false;
else
FIRSTCATCH = true;
if(FIRSTCATCH == false)
return false;
else
{
prop = makeEntityStringProperties(st);
return checkFilterAndProperties(prop, filters);
}
}
else
{
log.warn("Error in filtering: obj = " + obj);
//System.out.println("Error in filtering: obj = " + obj);
return false;
}
}

//Совпадение = true
private boolean checkFilterAndProperties(String prop, String[] filters)
{
int i;
if(filter == null || filter.trim().equals(""))
return true;
for(i = 0; i < filters.length; ++i)
if(prop.contains(filters[i]))
return true;
return false;
}

```

```

private static String makeEntityStringProperties(Object obj)
{
    String res;
    if(obj.getClass() == Employee.class)
    {
        Employee e = (Employee)obj;
        res = e.getID() + " " + e.getName() + " " + e.getGender() + " " +
        (e.getGender()==true?"M":"F") + " " + e.getPassport() + " " + e.getBirthday() + " " +
        Salon.doNiceDate(e.getBirthday()) + " ";
        Set<ServiceType> sts = e.getMasteredServices();
        for(ServiceType st : sts)
        res += st.getID() + " ";
        return res;
    }
    else if(obj.getClass() == Client.class)
    {
        Client c = (Client)obj;
        res = c.getID() + " " + c.getName() + " " + c.getGender() + " " +
        (c.getGender()==true?"M":"F") + " " + c.getPassport() + " " + c.getBirthday() + " " +
        Salon.doNiceDate(c.getBirthday()) + " ";
        res += c.getPriority();
        return res;
    }
    else if(obj.getClass() == Service.class)
    {
        Service s = (Service)obj;
        res = s.getID() + " " + s.getDateBegin() + " " + Salon.doNiceDate(s.getDateBegin()) + " " +
        s.getPrice() + " " + s.getCashReward() + " " + s.getServiceType().getDescription() + " - " +
        s.getServiceType().getID() + " " + s.getEmployee().getName() + " - " +
        s.getEmployee().getID() + " " + s.getClient().getName() + " - " + s.getClient().getID();
        return res;
    }
    else if(obj.getClass() == ServiceType.class)
    {
        ServiceType st = (ServiceType)obj;
        res = st.getID() + " " + st.getDescription() + " " + st.getCurrentPrice() + " " + st.getPercent();
        return res;
    }
    else
    {
        log.warn("Error in makeEntityStringProperties: obj = " + obj);
        //System.out.println("Error in makeEntityStringProperties: obj = " + obj);
        return null;
    }
}

```

```

private static Object[][] removeNull(Object[][] o, int m)
{
    int i, j;
    int countNull;
    countNull = 0;
    for(i = 0; i < o.length; ++i)
        if(o[i] != null)
            ++countNull;
    Object[][] res = new Object[countNull][];
    for(i = 0; i < res.length; ++i)
        res[i] = new Object[m];
    for(i = 0, j = 0; i < o.length; ++i)
        if(o[i] != null)
            res[j++] = o[i];
    return res;
}

```

```

private void editItem(JScrollPane cur)
{
    log.info("Edit button choosed.");

```

```

    int selRow = -1;
    int selID = -1;

```

```

    if(cur == employeePane)
    {
        log.info("Editing employees choosed.");

```

```

        selRow = employeeTable.getSelectedRow();
        if(selRow == -1)
        {
            JOptionPane.showMessageDialog(mainFrame, "First select the row");
            return;
        }

```

```

        selID = Integer.parseInt(employeeTable.getValueAt(selRow, 0).toString());

```

```

        log.info("Editing employee " + employeeTable.getValueAt(selRow, 1).toString() + " - " + selID
            + ".");

```

```

        if(employeeEditor == null)
            employeeEditor = new EditEmployee(mainFrame, salon.getEmployeeByID(selID));
        else
            employeeEditor.updateEdit(salon.getEmployeeByID(selID));
        employeeEditor.setVisible(true);
    }
    else if(cur == clientPane)
    {

```

```

log.info("Editing clients choosed.");

selRow = clientTable.getSelectedRow();
if(selRow == -1)
{
JOptionPane.showMessageDialog(mainFrame, "First select the row");
return;
}
selID = Integer.parseInt(clientTable.getValueAt(selRow, 0).toString());

log.info("Editing client " + clientTable.getValueAt(selRow, 1).toString() + " - " + selID + ".");

if(clientEditor == null)
clientEditor = new EditClient(mainFrame, salon.getClientByID(selID));
else
clientEditor.updateEdit(salon.getClientByID(selID));
clientEditor.setVisible(true);
}
else if(cur == servicePane)
{
log.info("Editing services choosed.");

selRow = serviceTable.getSelectedRow();
if(selRow == -1)
{
JOptionPane.showMessageDialog(mainFrame, "First select the row");
return;
}
selID = Integer.parseInt(serviceTable.getValueAt(selRow, 0).toString());

log.info("Editing service with ID = " + selID + ".");

if(serviceEditor == null)
serviceEditor = new EditService(mainFrame, salon.getServiceByID(selID));
else
serviceEditor.updateEdit(salon.getServiceByID(selID));
serviceEditor.setVisible(true);
}
else if(cur == serviceTypePane)
{
log.info("Editing service types choosed.");

selRow = serviceTypeTable.getSelectedRow();
if(selRow == -1)
{
JOptionPane.showMessageDialog(mainFrame, "First select the row");
return;
}

```



```

}
sellID = Integer.parseInt(serviceTypeTable.getValueAt(selRow, 0).toString());

log.info("Editing service type: " + serviceTypeTable.getValueAt(selRow, 1).toString() + " - " +
sellID + ".");

if(serviceTypeEditor == null)
serviceTypeEditor = new EditServiceType(mainFrame, salon.getServiceTypeByID(sellID));
else
serviceTypeEditor.updateEdit(salon.getServiceTypeByID(sellID));
serviceTypeEditor.setVisible(true);
}
else
{
log.warn("Error in editItem: cur = " + cur);
//System.out.println("Error in editItem: cur = " + cur);
}
//System.out.println("TunA u3MEHuJlOCb...");
}

private void addItem(JScrollPane cur)
{
log.info("Add button choosed.");

if(cur == employeePane)
{
log.info("Adding employee.");

if(employeeAdder == null)
employeeAdder = new AddEmployee(mainFrame);
else
employeeAdder.updateAdd();
employeeAdder.setVisible(true);

log.info("Employee added.");
}
else if(cur == clientPane)
{
log.info("Adding client.");

if(clientAdder == null)
clientAdder = new AddClient(mainFrame);
else
clientAdder.updateAdd();
clientAdder.setVisible(true);

log.info("Client added.");
}

```

```

}
else if(cur == servicePane)
{
log.info("Adding service.");

if(serviceAdder == null)
serviceAdder = new AddService(mainFrame);
else
serviceAdder.updateAdd();
serviceAdder.setVisible(true);

log.info("Service added.");
}
else if(cur == serviceTypePane)
{
log.info("Adding service type.");

if(serviceTypeAdder == null)
serviceTypeAdder = new AddServiceType(mainFrame);
else
serviceTypeAdder.updateAdd();
serviceTypeAdder.setVisible(true);

log.info("Service type added.");
}
else
{
log.warn("Error in addItem: cur = " + cur);
//System.out.println("Error in addItem: cur = " + cur);
}
//System.out.println("TunA DO6ABuJlOCb...");
}

private void doTask()
{
log.info("Task button choosed.");

if(taskHundler == null)
taskHundler = new TaskHundler(mainFrame);
else
taskHundler.update();
taskHundler.setVisible(true);
}
}

```

KY39.java

```

import java.lang.*;
import java.util.*;
import java.io.*;

import lab.*;
import labgui.*;

import org.apache.log4j.Logger;

/**
 * Класс, который всё запускает
 */
public class KY39
{
    private static final Logger log = Logger.getLogger(KY39.class);

    /**
     * Запуск
     */
    * @param args
    */
    public static void main(String[] args)
    {
        //> mvn compile exec:java -Dexec.mainClass="KY39"
        //> mvn test
        //> mvn javadoc:aggregate-jar
        log.info("Starting");
        SalonGUI.start();
    }
}

```

TaskHundler.java

```

package labgui;

import java.lang.*;
import java.util.*;
import java.awt.*;
import javax.swing.*;
import javax.swing.table.*;
import javax.swing.filechooser.FileFilter;
import javax.swing.filechooser.FileNameExtensionFilter;
import java.awt.event.*;
import java.io.*;

```

```

import lab.*;
import labgui.*;

import org.apache.log4j.Logger;
import org.apache.log4j.Level;

/**
 * Форма для составления статистики работы салона красоты и для составления от-
 * чётов
 */
class TaskHundler extends JDialog
{
    private static final Logger log = Logger.getLogger(TaskHundler.class);

    private final int DEFAULT_WIDTH = 1280;
    private final int DEFAULT_HEIGHT = 720;

    private SalonGUI salonGUI;
    private Salon salon;

    private JPanel mainPanel;
    private JPanel tablePanel;
    private JPanel upPanel;
    private JPanel downPanel;

    private JButton testB;
    private DateChooser dc;

    private Employee choosedEmployeeWorkload;
    private Employee choosedEmployeeRecord;
    private Client choosedClientRecord;

    private boolean REPORTING;

    private boolean READY_bool;

    Set<Service> lastChoosedServices;

    /*Day = 0
    Week = 1
    Month = 2*/
    private int D_W_M;

    public TaskHundler(JFrame owner)
    {
        super(owner, "Task", true);
    }

```

```

log.info("Creating new task frame...");

mainPanel = new JPanel();
tablePanel = new JPanel();

salonGUI = SalonGUI.getSalonGUI();
salon = salonGUI.getSalon();

mainPanel.setLayout(new BorderLayout());
tablePanel.setLayout(new BorderLayout());
upPanel = new JPanel();

update();

downPanel = new JPanel();

mainPanel.add(upPanel, BorderLayout.NORTH);
mainPanel.add(tablePanel, BorderLayout.CENTER);
mainPanel.add(downPanel, BorderLayout.SOUTH);

this.add(mainPanel);
this.setLocationRelativeTo(owner);
pack();
}

/**
 * Обновить содержимое формы
 *
 */
public void update()
{
log.info("Updating all components on task frame.");

READY_bool = false;

upPanel.removeAll();

prepareRasp(upPanel);

prepareEmployeeRasp(upPanel);

prepareClientRasp(upPanel);

prepare_D_W_M(upPanel);

dc = new DateChooser();

```

```
upPanel.add(dc);
```

```
prepareReportButton(upPanel);
```

```
upPanel.updateUI();
```

```
}
```

```
public Dimension getPreferredSize()
```

```
{
```

```
return new Dimension(DEFAULT_WIDTH, DEFAULT_HEIGHT);
```

```
}
```

```
private void prepareRasp(JPanel upPanel)
```

```
{
```

```
 JButton showRaspButton = new JButton("Show schedule");
```

```
showRaspButton.addActionListener(event ->
```

```
{
```

```
if(D_W_M == 0)
```

```
makeTable( salon.showServicesPerDay(dc.getDate()) );
```

```
else if(D_W_M == 1)
```

```
makeTable( salon.showServicesPerWeek(dc.getDate()) );
```

```
else if(D_W_M == 2)
```

```
makeTable( salon.showServicesPerMonth(dc.getDate()) );
```

```
});
```

```
upPanel.add(showRaspButton);
```

```
}
```

```
private void prepareEmployeeRasp(JPanel upPanel)
```

```
{
```

```
 JButton showRaspButton = new JButton("Employee's workload");
```

```
showRaspButton.addActionListener(event ->
```

```
{
```

```
if(choosedEmployeeWorkload != null)
```

```
{
```

```
if(D_W_M == 0)
```

```
makeTable( salon.calculateWorkloadDay(dc.getDate(), choosedEmployeeWorkload) );
```

```
else if(D_W_M == 1)
```

```
makeTable( salon.calculateWorkloadWeek(dc.getDate(), choosedEmployeeWorkload) );
```

```
else if(D_W_M == 2)
```

```
makeTable( salon.calculateWorkloadMonth(dc.getDate(), choosedEmployeeWorkload) );
```

```
}
```

```
});
```

```
String[] eComboStr = prepareEmployeeList(salon.getEmployees());
```

```

JComboBox eCombo = new JComboBox<String>(eComboStr);
ActionListener eComboActionListener = new ActionListener() {
public void actionPerformed(ActionEvent e) {
JComboBox box = (JComboBox)e.getSource();
choosedEmployeeWorkload = salon.getEmployeeByID(
takeIDfromCombo((String)box.getSelectedItem()) );
}
};
eCombo.addActionListener(eComboActionListener);

```

```

JPanel eRaspPanel = new JPanel();
eRaspPanel.setLayout(new GridLayout(2, 1));

```

```

eRaspPanel.add(showRaspButton);
eRaspPanel.add(eCombo);
upPanel.add(eRaspPanel);
}

```

```

private void prepareClientRasp(JPanel upPanel)
{
Button showRaspButton = new Button("Specialist\'s appointment");

```

```

showRaspButton.addActionListener(event ->
{
if(choosedEmployeeRecord != null && choosedClientRecord != null)
{
if(D_W_M == 0)
makeTable( salon.calculateClientRaspDay(dc.getDate(), choosedEmployeeRecord,
choosedClientRecord) );
else if(D_W_M == 1)
makeTable( salon.calculateClientRaspWeek(dc.getDate(), choosedEmployeeRecord,
choosedClientRecord) );
else if(D_W_M == 2)
makeTable( salon.calculateClientRaspMonth(dc.getDate(), choosedEmployeeRecord,
choosedClientRecord) );
}
});

```

```

String[] eComboStr = prepareEmployeeList(salon.getEmployees());
JComboBox eCombo = new JComboBox<String>(eComboStr);
ActionListener eComboActionListener = new ActionListener() {
public void actionPerformed(ActionEvent e) {
JComboBox box = (JComboBox)e.getSource();
choosedEmployeeRecord = salon.getEmployeeByID(
takeIDfromCombo((String)box.getSelectedItem()) );
}
};

```

```
eCombo.addActionListener(eComboActionListener);
```

```
String[] cComboStr = prepareClientList(salon.getClients());  
JComboBox cCombo = new JComboBox<String>(cComboStr);  
ActionListener cComboActionListener = new ActionListener() {  
    public void actionPerformed(ActionEvent e) {  
        JComboBox box = (JComboBox)e.getSource();  
        choosedClientRecord = salon.getClientByID( takeIDfromCombo((String)box.getSelectedItem()  
    );  
    }  
};  
cCombo.addActionListener(cComboActionListener);
```

```
JPanel ecRaspPanel = new JPanel();  
ecRaspPanel.setLayout(new GridLayout(3, 1));
```

```
ecRaspPanel.add(showRaspButton);  
ecRaspPanel.add(eCombo);  
ecRaspPanel.add(cCombo);  
upPanel.add(ecRaspPanel);  
}
```

```
private static String[] prepareEmployeeList(Set<Employee> es)  
{  
    int i;  
    String[] res = new String[es.size()];  
    i = 0;  
    for(Employee e : es)  
        res[i++] = e.getName() + " - " + e.getID();  
    Arrays.sort(res, (String a, String b) -> {return a.compareTo(b);});  
  
    return res;  
}
```

```
private static String[] prepareClientList(Set<Client> cs)  
{  
    int i;  
    String[] res = new String[cs.size()];  
    i = 0;  
    for(Client c : cs)  
        res[i++] = c.getName() + " - " + c.getID();  
    Arrays.sort(res, (String a, String b) -> {return a.compareTo(b);});  
  
    return res;  
}
```

```
private static int takeIDfromCombo(String s)
```



```

{
int res;
int i = s.lastIndexOf(" - ");
String buffS = s.substring(i + 3, s.length());
res = Integer.parseInt(buffS);
return res;
}

private void prepare_D_W_M(JPanel upPanel)
{
JPanel RatioPanel = new JPanel();
RatioPanel.setLayout(new GridLayout(3, 1));

JRadioButton MonthRatio = new JRadioButton("Month");
JRadioButton WeekRatio = new JRadioButton("Week");
JRadioButton DayRatio = new JRadioButton("Day");

ActionListener dwmListener = ae ->
{
if(MonthRatio.isSelected())
D_W_M = 2;
else if(WeekRatio.isSelected())
D_W_M = 1;
else if(DayRatio.isSelected())
D_W_M = 0;
};

MonthRatio.addActionListener(dwmListener);
WeekRatio.addActionListener(dwmListener);
DayRatio.addActionListener(dwmListener);

MonthRatio.setSelected(true);
D_W_M = 2;

ButtonGroup dwmRatioGroup = new ButtonGroup();
dwmRatioGroup.add(MonthRatio);
dwmRatioGroup.add(WeekRatio);
dwmRatioGroup.add(DayRatio);

RatioPanel.add(MonthRatio);
RatioPanel.add(WeekRatio);
RatioPanel.add(DayRatio);

upPanel.add(RatioPanel);
}

private void prepareReportButton(JPanel upPanel)

```

```

{
JButton reportButton = new JButton(new ImageIcon(SalonGUI.class.getResource("/img/
pdfButton.png")));
reportButton.setToolTipText("Make report");
REPORTING = false;
reportButton.addActionListener(event ->
{
if(REPORTING == false)
{
log.info("Begin do OT4ET PDF");
if(READY_bool == true)
{
REPORTING = true;
JFileChooser fileChooser = new JFileChooser();
fileChooser.setDialogTitle("Select where you want to save the report");
FileFilter pdf = new FileNameExtensionFilter("PDF file(.pdf)", "pdf");
fileChooser.addChoosableFileFilter(pdf);
fileChooser.setCurrentDirectory(new File("."));
int choice = fileChooser.showSaveDialog(this);
if(choice == JFileChooser.APPROVE_OPTION)
{
File file = fileChooser.getSelectedFile();
String OT4ET_PDF_path = file.getAbsolutePath();
Runnable task = () ->
{
ReportMaker.makeReport(salon, lastChoosedServices, D_W_M, dc.getDate(),
OT4ET_PDF_path/"./Report.pdf");
};
new Thread(task).start();

JOptionPane.showMessageDialog(this, "OT4ET PDF Generated", "Generated",
JOptionPane.INFORMATION_MESSAGE);
}

REPORTING = false;
}
else
JOptionPane.showMessageDialog(this, "First create the tables for the report", "Error",
JOptionPane.ERROR_MESSAGE);

}
});
upPanel.add(reportButton);
}

private void makeTable(Set<Service> services)
{

```

```
log.info("Preparing task table...");
```

```
lastChoosedServices = services;
```

```
tablePanel.removeAll();
```

```
String[] servicesCol = {"ID", "Service type", "Date", "Employee", "Employee\'s cut", "Client",  
"Price", "Relevance"};
```

```
Object[][] servicesData = services2Table(services);
```

```
DefaultTableModel serviceModel = new DefaultTableModel(servicesData, servicesCol)
```

```
{
```

```
@Override
```

```
public boolean isCellEditable(int i, int i1) {
```

```
    return false;
```

```
}
```

```
@Override
```

```
public Class getColumnClass(int column)
```

```
{
```

```
    if(column == 0)
```

```
        return Integer.class;
```

```
    else if(column == 4 || column == 6)
```

```
        return Double.class;
```

```
    else
```

```
        return String.class;
```

```
}
```

```
};
```

```
JTable serviceTable = new JTable(serviceModel);
```

```
serviceTable.setAutoCreateRowSorter(true);
```

```
serviceTable.setEnabled(true);
```

```
DefaultTableCellRenderer r = (DefaultTableCellRenderer)
```

```
serviceTable.getDefaultRenderer(Integer.class);
```

```
r.setHorizontalAlignment(JLabel.LEFT);
```

```
serviceTable.getTableHeader().setReorderingAllowed(false);
```

```
tablePanel.add(new JScrollPane(serviceTable));
```

```
tablePanel.updateUI();
```

```
makeStatistics(downPanel);
```

```
READY_bool = true;
```

```
log.info("Task table was prepared.");
```

```
}
```

```
private void makeStatistics(JPanel downPanel)
```

```

{
log.info("Making statistics...");

downPanel.removeAll();

//=====================================================money=====
JPanel income = new JPanel();
income.setLayout(new GridLayout(2, 1));

double inWithoutPer = 0;
double inWithPer = 0;
if(D_W_M == 0)
{
inWithoutPer = salon.calculateIncomeCashDay(dc.getDate(), false);
inWithPer = salon.calculateIncomeCashDay(dc.getDate(), true);
}
else if(D_W_M == 1)
{
inWithoutPer = salon.calculateIncomeCashWeek(dc.getDate(), false);
inWithPer = salon.calculateIncomeCashWeek(dc.getDate(), true);
}
else if(D_W_M == 2)
{
inWithoutPer = salon.calculateIncomeCashMonth(dc.getDate(), false);
inWithPer = salon.calculateIncomeCashMonth(dc.getDate(), true);
}

income.add(new JLabel("Income: " + inWithoutPer));
income.add(new JLabel("With percent: " + inWithPer));
downPanel.add(income);

//=====================================================number of clients=====
JPanel number = new JPanel();
number.setLayout(new GridLayout(2, 1));

int unum = 0;
int num = 0;
if(D_W_M == 0)
{
unum = salon.calculateClientsNumDay(dc.getDate(), true);
num = salon.calculateClientsNumDay(dc.getDate(), false);
}
else if(D_W_M == 1)
{
unum = salon.calculateClientsNumWeek(dc.getDate(), true);
num = salon.calculateClientsNumWeek(dc.getDate(), false);
}
}

```

```

else if(D_W_M == 2)
{
    unum = salon.calculateClientsNumMonth(dc.getDate(), true);
    num = salon.calculateClientsNumMonth(dc.getDate(), false);
}

```

```

number.add(new JLabel("Number of clients: " + num));
number.add(new JLabel("Unique: " + unum));
downPanel.add(number);

```

```

//=====Best=====

```

```

JPanel bests = new JPanel();
bests.setLayout(new GridLayout(2, 1));

```

```

Employee eBestMoney = null;
Employee eBestTraffic = null;

```

```

if(D_W_M == 0)
{
    eBestMoney = salon.cal_BestCashEmployee_Day(dc.getDate());
    eBestTraffic = salon.cal_BestTrafficEmployee_Day(dc.getDate());
}
else if(D_W_M == 1)
{
    eBestMoney = salon.cal_BestCashEmployee_Week(dc.getDate());
    eBestTraffic = salon.cal_BestTrafficEmployee_Week(dc.getDate());
}
else if(D_W_M == 2)
{
    eBestMoney = salon.cal_BestCashEmployee_Month(dc.getDate());
    eBestTraffic = salon.cal_BestTrafficEmployee_Month(dc.getDate());
}

```

```

bests.add(new JLabel("Best money: " + (eBestMoney == null?"None":eBestMoney.getName()
+ " - " + eBestMoney.getID() ));
bests.add(new JLabel("Best traffic: " + (eBestTraffic == null?"None":eBestTraffic.getName() +
" - " + eBestTraffic.getID() ));
downPanel.add(bests);

```

```

downPanel.updateUI();

```

```

log.info("Statistics was formed");
}

```

```

private Object[][] services2Table(Set<Service> ss)
{
    int i;

```

```

int num = 8;
Object[][] res = new Object[ss.size()][8];
for(i = 0; i < res.length; ++i)
res[i] = new Object[num];
i = 0;
for(Service s : ss)
{
res[i][0] = Integer.valueOf(s.getID());
res[i][1] = s.getServiceType().getDescription() + " - " + s.getServiceType().getID();
res[i][2] = Salon.doOnlyDate(s.getDateBegin());
res[i][3] = s.getEmployee().getName() + " - " + s.getEmployee().getID();
res[i][4] = Double.valueOf(s.getCashReward());
res[i][5] = s.getClient().getName() + " - " + s.getClient().getID();
res[i][6] = Double.valueOf(s.getPrice());
res[i][7] = "" + s.isRELEVANT();
++i;
}
return res;
}
}

```

hibernate.cfg.xml

```

<!DOCTYPE hibernate-configuration PUBLIC
"-//Hibernate/Hibernate Configuration DTD 3.0//EN"
"http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
<session-factory>
<!-- JDBC Database connection settings -->
<property name="connection.driver_class">com.mysql.cj.jdbc.Driver</property>
<property name="connection.url">jdbc:mysql://localhost:1234/lab</property>
<property name="connection.username">LABUSER</property>
<property name="connection.password">SECRETPASSWORD.IDONTTELL!</property>
<!-- JDBC connection pool settings ... using built-in test pool -->
<property name="connection.pool_size">1</property>
<!-- Select our SQL dialect -->
<property name="dialect">org.hibernate.dialect.MySQLDialect</property>
<!-- Echo the SQL to stdout -->
<property name="show_sql">>false</property>
<!-- Set the current session context -->
<property name="current_session_context_class">thread</property>
</session-factory>
</hibernate-configuration>

```

log4j.properties

```
log =.  
log4j.rootLogger = INFO, FILE  
log4j.appender.FILE=org.apache.log4j.FileAppender  
log4j.appender.FILE.File=${log}/log.out  
log4j.appender.FILE.Encoding=UTF-8  
log4j.appender.FILE.layout=org.apache.log4j.PatternLayout  
log4j.appender.FILE.layout.conversionPattern=[%r is %d{yyyy-MM-dd HH:mm:ss,SSS} %-5p  
%c:%M:%L] %m%n
```