



МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«МИРЭА – Российский технологический университет»

РТУ МИРЭА

Институт кибербезопасности и цифровых технологий

Кафедра КБ-14 «Цифровые технологии обработки данных»

Кроссплатформенная среда исполнения программного обеспечения

Практическая работа №1

Основы синтаксиса и объектно-ориентированного программирования

Вводная

Первая практическая работа ориентирована прежде всего на освоение синтаксиса языка Java. Не смотря на схожесть языка C# с Java, имеет смысл привыкнуть как к особенностям конкретно этого языка, так и инструментарию. Тем, кто работал с Rider от JetBrains будет проще освоится с IntelliJ, чем после Visual Studio (с другой стороны, программировать на Java можно хоть в блокноте, только компиляция и запуск программы потребует некоторых ухищрений).

Если читатель данных строк работал ранее только с Python или C/C++, то освоение синтаксиса потребует большего времени засчёт иной парадигмы программирования и стиля написания кода. В любом случае, чтобы понимание синтаксиса не стало болью, предусмотрено специальное дополнение к первой лекции, где в понятных примерах объясняется основы синтаксиса.

Второе назначение данной практики – освоение объектно-ориентированного программирования с помощью разработки небольшого приложения на основе диаграммы UML. Такое задание позволит лучше понять принципы ООП, которые являются необходимыми для написания Java приложений. Помимо этого, изучаются некоторые элементы языка, такие как лямбда-функции, коллекции, исключения и т.п.

Задание 1

На вход программы приходит сообщение закодированное азбукой Морзе. Каждая буква сообщения отделяется одним пробелом, а слова – переносом строки. Например, слово “привет мир” будет выглядеть так:

```
.-.-. .-. ... .-- . -  
-- .. .-.
```

Программа расшифровывает сообщение и выводит итоговый текст в консоль.

Требования к решению задания:

- Загрузка алфавита из текстового файла в словарь типа Map.
- Для разбиения входного сообщения на отдельные элементы использовать встроенные функции типа String.
- Реализовывать знаки пунктуации не требуется.

Задание 2

На некотором складе разрабатывается автоматизированная система хранения вещей. По проекту существует интерфейс `Box<T>`, описывающий некоторую коробку в которую можно положить вещь, открыть и посмотреть содержимое или забрать её.

Соответственно, методы интерфейса выглядят следующим образом:

```
T open();  
put(T item);  
T removeFromBox();
```

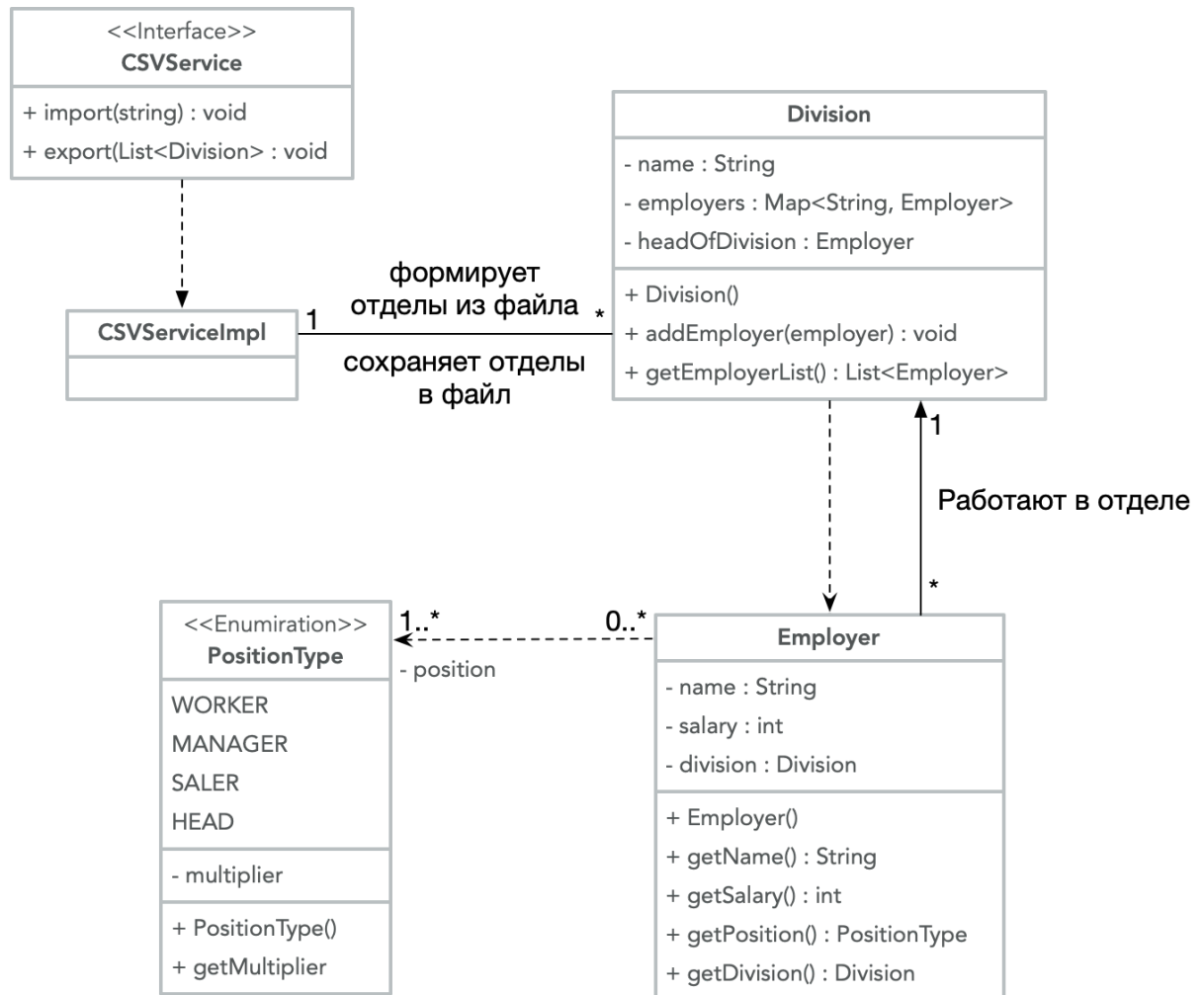
Необходимо реализовать интерфейс `Box` в классах, которые бы описывали:

- обувная коробка;
- книжная коробка;
- пивная коробка.

Соответственно, для этого надо создать классы `Обувь`, `Книга` и `Пиво`.

Задание 3

Реализуйте классы, представленные на данной UML-диаграмме:



Задание 4

На основе реализованных классов из предыдущего задания разработайте консольное приложение - систему управления сотрудниками и отделами компании. Система должна поддерживать хранение и организацию информации о сотрудниках, включая их имя, должность, зарплату и отдел. Она также должна иметь возможность импортировать и экспортировать информацию о сотрудниках в виде файла CSV.

Задачи:

1. Реализуйте метод для импорта информации о сотрудниках из CSV-файла. В файле должна быть строка для каждого сотрудника в следующем формате:
“ФИО, должность, зарплата, отдел”
2. Реализуйте метод для экспорта информации о сотрудниках в файл CSV в том же формате, что и выше.
3. Протестируйте готовую систему следующим образом:
 - i. Создайте три отдела с разными названиями.
 - ii. Добавьте какое-то количество сотрудников с разной зарплатой, которая зависит от должности (множитель `multiplier` в `enum PositionType`)
 - iii. При добавлении сотрудника, если его должность является `HEAD`, то он автоматически добавляется в поле `headOfDivision`.
 - iv. Сохранить список сотрудников в формат CSV файла.
 - v. Запустить программу и импортировать структуру компании из CSV файла.
 - vi. Вывести следующую информацию о каждом отделе: глава отдела, его зарплата, количество сотрудников, их средняя зарплата.

Что должно быть в отчёте

В отчёте должно быть следующее:

1. текст задания;
2. листинг кода по каждому заданию;
3. скриншоты работы программы (консольного окна).

Оформление отчёта

Обратите внимание на то, что отчёт перед сдачей на проверку должен быть оформлен согласно следующим требованиям:

- Основной шрифт для текста - Times New Roman, 14 пт;
- Межстрочный интервал - 1,5;
- Выравнивание - по ширине;
- Шрифт для листинга кода, названия классов и т.д - Courier New, Consolas или JetBrains Mono, размер не более 12 пт;
- Листинг кода должен быть на белом фоне (никаких скриншотов); форматирование можно осуществлять через плагин Easy Code Formatter (для MS Word 2016 и выше), либо через сервис <https://pastebin.com/>
 - Схема для Pastebin такая:
 1. Вставляете код класса из IDE;
 2. В настройках снизу указываете:
 - a. Syntax Highlighting: **Java**,
 - b. Paste Expiration: **Burn after read**,
 - c. Paste Exposure: **Unlisted**.
 3. Нажимаете Create new paste;
 4. Копируете полученный код и вставляете в документ.
- Оформление титульного листа должно быть по единой утверждённой форме (брать у старост).
 - На данный момент название института: “Институт кибербезопасности и цифровых технологий”.

Не оформленный должным образом отчёт приниматься на проверку не будет.

Рекомендации по выполнению практической работы

Основные материалы для помощи

При выполнении работ, возможно, потребуется ознакомление с общим синтаксисом языка, для этого доступна лекция: [Лекция 1.5 Базовый синтаксис Java](#) (СДО)

Для выполнения заданий 1 и 4 потребуется информация из следующих лекций:

[Лекция 2.6 Работа с файлами](#) (СДО)

[Лекция 2.4 Тип данных enum](#) (СДО)

Для задания 3 полезной будет:

[Лекция 2.2 UML](#) (СДО)

Для задания 2 :

[Лекция 2. Объектно-ориентированный подход к написанию программ](#) (страница 23-24, а также 32).

Помимо этого, подавляющее большинство информации для помощи при выполнении практической работы находится в учебно-методическом пособии [по ссылке из онлайн-библиотеки](#):

1. Основы ООП – 20 стр.
2. enum – 30 стр.
3. Дженирики – 36 стр.

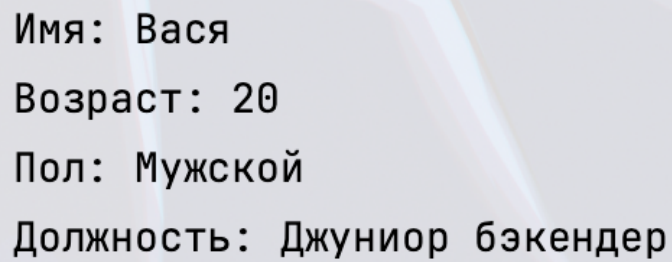
Вывод информации о классе

Для удобства получения данных из свойств класса можно воспользоваться переопределением стандартного метода `toString()`, который присутствует во всех классах, так как неявно наследуется от супер-класса `Object`.

Например, у нас есть некоторый класс `Person`, о котором мы хотим получить данные об имени, пол, возраст и должность.

```
@Override
public String toString() {
    return "Имя: "+this.name+"\n"+
           "Возраст: "+this.age+"\n"+
           "Пол: "+this.gender+"\n"+
           "Должность: "+this.position;
}
```

Метод возвращает объект строки, которая на выводе в консоль будет разделена на четыре строки с помощью спец.символа "\n".



```
Имя: Вася
Возраст: 20
Пол: Мужской
Должность: Джуниор бэкендер
```