Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Лабораторна робота №8

# Технології розроблення програмного забезпечення

*ШАБЛОНИ «COMPOSITE», «FLYWEIGHT», «INTERPRETER», «VISITOR»*

Варіант №14

Виконав:

студент групи ІА-14,

Міщук Максим Дмитрович

Перевірив:

Мягкий М.Ю.

Київ 2023

**Тема роботи:** Шаблони «Composite», «Flyweight», «Interpreter», «Visitor».

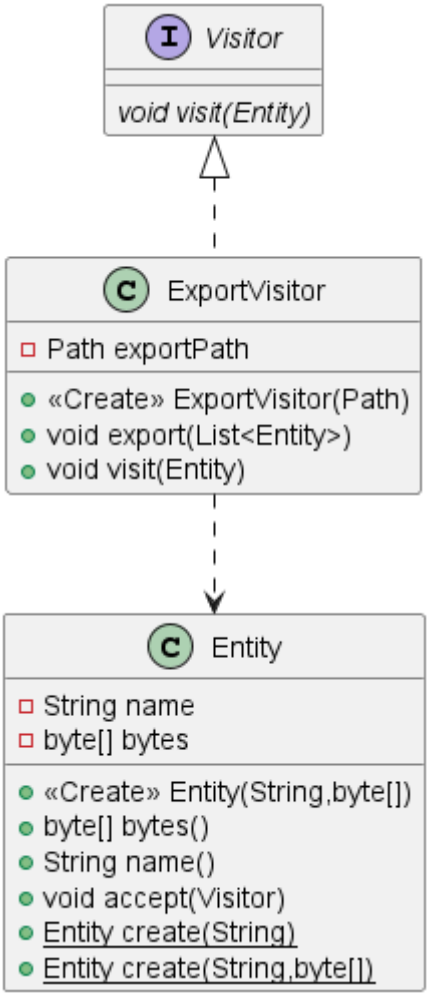
**Вхідні дані:**

**..14 Архіватор (strategy, adapter, factory method, facade, visitor, p2p)**

Архіватор повинен являти собою візуальний додаток з можливістю створення і редагування архівів різного типу (.tar.gz, .zip, .rar, .ace) - додавання/ видалення файлів / папок, редагування метаданих (по можливості), перевірка checksum архівів, тестування архівів на наявність пошкоджень, розбиття архівів на частини.

**Хід роботи:**

На основі шаблону відвідувач було зроблене розпакування файлів з архів. Кожна сутність архіву має в собі метод що приймає «відвідувача» і цей відвідувач викликає свій функціонал над даною сутністю. Відвідувач зчитує дані сутності та записує їх за заданим шляхом користувачем.




*Діаграма класів*

Відвідувач має в собі лише один метод, котрий приймає в себе об’єкт класу Entity. В майбутньому можна додати типізацію до даного інтерфейсу, щоб можна було його використовувати з іншими класами. Далі перейдемо до єдиної реалізації:

При ініціалізації об’єкту конструктор класу приймає в себе шлях, куди буде розпаковано архів. Далі було створено метод *export()*, що приймає в себе список сутностей. В середині він починає по черзі викликати метод *accept()* у сутностей і, як параметр, передавати посилання на самого себе. Як було описано раніше в середині сутності викликається метод *visit()* та в якості параметру передається посилання на сутність.

В самому методі *visit()* на початку поєднується шлях експортування із внутрішнім шляхом функції. Після чого створюються директорії ієрархії (за необхідності). Якщо за отриманим шляхом вже існує файл, то викликається утилітарний метод *getFreePath()* класу *FileUtils*, що видозмінює посилання для можливості зберегти файл без конфліктів та перезапису вже існуючого.

Код реалізації:

```java
public interface Visitor {
    void visit(Entity entity);
}
```

*Visitor.java*

```java
public class ExportVisitor implements Visitor {
    private final Path exportPath;

    public ExportVisitor(Path exportPath) {
        if (!Files.isDirectory(exportPath)) {
            throw new IllegalArgumentException("Path must be a directory!");
        }
        this.exportPath = exportPath;
    }

    public void export(List<Entity> entityList) {
        entityList.forEach(entity -> entity.accept(this));
    }

    @Override
    public void visit(Entity entity) {
        Path entityPath = exportPath.resolve(entity.name());
        Path parentPath = entityPath.getParent();
        try {
            Files.createDirectories(parentPath);
            entityPath = FileUtils.getFreePath(entityPath);

            try (OutputStream os = Files.newOutputStream(entityPath)) {
                os.write(entity.bytes());
            }

            Files.setLastModifiedTime(
                    entityPath,
                    FileTime.from(entity.lastModifiedDate().toInstant())
            );
        }
        catch (IOException ex) {
            throw new RuntimeException(ex);
        }
    }
}
```

*ExportVisitor.java*

```java
public class FileUtils {
    public static Path getFreePath(Path path) {
        Path parentPath = path.getParent();

        if (Files.exists(path)) {
            String name = FileNameUtils.getBaseName(path);
            StringBuilder extension = new StringBuilder();
            extension.insert(0, FileNameUtils.getExtension(path));

            while (!FileNameUtils.getExtension(name).isBlank()) {
                extension.insert(0, FileNameUtils.getExtension(name) + ".");
                name = FileNameUtils.getBaseName(name);
            }


            for (int i = 1; Files.exists(path); i++) {
                path = parentPath.resolve(name + "(" + i + ")" + "." + extension);
            }
        }

        return path;
    }
}
```

*FileUtils.java*

**Висновок:**

Використання шаблону відвідувач, дало змогу зробити функцію розпакування як додаткову, заснована на базових функціях описаних в першому підрозділі та слугує гарним прикладом того як можна робити більш складну логіку на основі вже написаної без зміни попереднього коду.

Посилання на репозиторій: [Maxim-Mishchuk/trpz-archiver at develop (github.com)]