Цель – рассмотреть методы защиты файлового архива и обеспечить его защищенность.

Материал и методы. Для выполнения данного проекта используется операционная система Windows, исходя из рейтинга устанавливаемых операционных систем на персональные и корпоративные компьютеры, а так же программное обеспечение для шифрования TrueCrypt версии 7.1.1.0.

Результаты и их обсуждение. Анализ задач операционной системы, и требования предъявляемые к операционным системам (ОС) предполагает оценку эффективности, живучести, масштабируемости, расширяемости, мобильности, защищенности, интерактивности и практичности [2].

Проблемы информационной безопасности немыслимы без анализа средств защиты информации. Для защита файлового архива используется метод шифрования, с использованием утилиты TrueCrypt, а так же защита шифрованного контейнера от удаления и изменения с помощью разработанного программного обеспечения (ПО).

Стандартные средства защиты информации операционной системы (на примере Windows) [1]:

- Брандмауэр;
- резервное копирование;
- защитник Windows (Windows Defender);
- шифрование файлового архива;
- разграничение доступа;
- UAC (Контроль Учетных Записей);
- родительский контроль;
- SmartScreen;
- контроль сетевого трафика.

Так, в частности, шифрование файлового архива возможно с помощью не стандартного программного обеспечения TrueCrypt . TrueCrypt – программное обеспечение предназначенное для шифрования 32 и 64-разрядных операционных систем. Программа позволяет создавать зашифрованный логический (виртуальный) диск, хранящийся в виде файла (контейнера). Но тем не менее контейнер не защищен от удаления и изменения.

Для защиты шифрованного контейнера от удаления и изменения предлагается использовать дополнительное программное обеспечение. С этой целью предполагается разработка ПО, выполняющее функцию зашиты шифрованного контейнера – которое не реализовано в стандартных средствах защиты информации в операционных системах.

Заключение. Анализ средств защиты информации показывает, что для защиты файлового архива, не достаточно стандартных методов защиты информации. В связи с этим предлагается обеспечить безопасность файлового архива с применением программного обеспечения TrueCrypt, а защита шифрованного контейнера требует разработки дополнительного программного обеспечения.

Литература

1. Руссинович М., Соломон Д., Ионеску А. Р89 Внутреннее устройство Microsoft Windows. 6-е изд. Основные подсистемы ОС. – СПб.: Питер, 2014. – 672 с.
2. Иртегов Д.В. Введение в операционные системы. – СПб.: БХВ. – Петербург, 2002. – 624 с.
3. TrueCrypt – [Электронный ресурс] – Режим доступа: http://www.kap-yar.ru/index.php?pg=120

ПРИНЦИП ЕДИНСТВЕННОЙ ОТВЕТСТВЕННОСТИ И ЕГО ПРИМЕНЕНИЕ С ПОМОЩЬЮ ОБОБЩЕНИЙ И ИНТРОСПЕКЦИИ В ЯЗЫКЕ ПРОГРАММИРОВАНИЯ JAVA

Маркова А.А.

студентка 2 курса ВГУ имени П.М. Машерова, г. Витебск, Республика Беларусь

Научный руководитель – Ермоченко С.А., канд. физ.-мат. наук

Принцип единственной ответственности – один из основополагающих принципов объектно-ориентированного проектирования, который позволяет создавать программное обеспечение с легко изменяющейся структурой. Это очень важно в условиях работы с постоянно изменяющимися требованиями, что в последнее время становится всё более актуальной проблемой [1].

Сам принцип единственной ответственности говорит о том, что у класса должна существовать только одна причина его модификации, то есть должно существовать только одно требование, изменение которого может повлечь изменение класса [2]. Но такая формулировка данного принципа тяжело применима на практике, поэтому чаще всего используют следующий критерий: каждый класс должен быть ответственен за решение только одной, достаточно узкой, задачи.

Актуальность работы продиктована большим интересом к объектно-ориентированному проектированию при разработке программного обеспечения и известным проблемам применения данной парадигмы программирования.

Целью данной работы является рассмотрение вопроса о том, каким образом можно соблюсти принцип единственной ответственности с применением механизмов обобщений (Generics) и интроспекции (Reflection API).

Материал и методы. Исследование проводится на базе простой задачи отрисовки различных графических фигур на форме приложения. Эта задача достаточно проста, но в тоже время позволяет продемонстрировать предлагаемые здесь подходы. В качестве методов исследования использованы общенаучные методы анализа, сравнения и сопоставления, а также объектно-ориентированный анализ и проектирование.

Результаты и их обсуждение. В первую очередь рассмотрим, каким образом можно решить поставленную задачу с применением базовых принципов объектно-ориентированного программирования (наследования и полиморфизма). Для этого опишем абстрактный класс для хранения информации о фигурах (см. листинг 1). В данном примере класс нарушает принцип единственной ответственности, так как отвечает за хранение информации о фигуре (поля `x` и `y`), а также за отрисовку фигуры на форме (метод `paint`).

Листинг 1. Пример нарушения принципа единственной ответственности

```
abstract public class Figure {
 private int x, y;
 /* get-теры и set-теры */
 abstract public void paint(Graphics g);
}
```

Результатом нарушения принципа единственной ответственности могут быть проблемы при изменении способа отрисовки фигур. В самом деле, допустим, в приложении будет создано несколько десятков подклассов данного класса (для различных фигур). Если после этого необходимо будет поменять способ отрисовки фигур (например, в web-приложении), для этого придётся изменить несколько десятков классов. В такой ситуации необходимо не изменять существующие классы, а добавлять новые классы. Поэтому рассмотрим вариант реализации данного класса с разделением ответственности. За хранение информации о фигурах будет отвечать иерархия классов с классом `Figure` в вершине иерархии:

Листинг 2.

```
abstract public class Figure {
 private int x, y;
 /* get-теры и set-теры */
}
```

А за отрисовку фигур будет отвечать интерфейс FigureDrawer:

Листинг 3.

```
public interface FigureDrawer {
 void paint(Graphics g, Figure figure);
}
```

Но в таком виде работать с этим интерфейсом будет неудобно, так как в подклассах при реализации метода `paint` информацию о фигуре получить будет сложно, так как в метод передаётся только базовый класс `Figure`, и становится не ясно, какая конкретная фигура используется. Решить проблему можно с помощью явного приведения типов, что потенциально может порождать ещё большее число проблем [2].

Воспользуемся в данном случае механизмом обобщений [3]:

Листинг 3.

```
public interface FigureDrawer<T extends Figure> {
 void paint(Graphics g, T figure);
}
```

При реализации такого интерфейса можно указывать, какой конкретно фигурой параметризован обобщённый интерфейс (см. листинг 4). В этом листинге представлен пример для прямоугольника. Особое внимание необходимо обратить на второй параметр метода `paint`. В классе `RectangleDrawer` тип этого параметра уже не `Figure`, а `Rectangle`.

Листинг 4.

```
public class RectangleDrawer
 implements FigureDrawer<Rectangle> {
 @Override
 public void paint(Graphics g, Rectangle rectangle) {
 /* реализация метода */
 }
}
```

Остаётся открытым ещё один вопрос, каким образом теперь можно использовать иерархию классов, отвечающих за отрисовку классов. Раньше на форме, имея ссылку типа `Figure` можно было отри-

совать эту фигуру, вызвав непосредственно у этого объекта метод `paint`, при этом вовсе не нужно было знать, какой объект стоит за этой ссылкой.

Для решения этого вопроса воспользуемся возможностями интроспекции языка программирования Java (Reflection API) [3]:

Листинг 5.

```
public class DrawerCreator {
 private static Map<Class<? extends Figure>,
 FigureDrawer<? extends Figure>>
 drawers = new HashMap<>();
 @SuppressWarnings("unchecked")
 public static <T extends Figure> FigureDrawer<T>
 create(Class<T> figureClass) {
 return (FigureDrawer<T>)drawers.get(figureClass);
 }
}
```

Данный класс помогает получить объект класса для отрисовки конкретной фигуры, работая лишь на уровне абстракций.

Заключение. В данной работе предложена методика использования механизма обобщений и интроспекции для реализации принципов объектно-ориентированного проектирования, что позволяет создавать более гибкие и универсальные решения.

Литература

1. Hood, C., Wiedemann, S., Fichtinger, S., Pautz, U. Requirements Management: The Interface Between Requirements Development and All Other Systems Engineering Processes. – Springer-Verlag Berlin Heidelberg, 2008. – 275 p.
2. Мартин, Р.К., Ньюкирк, Дж. В., Косс, Р. С. Быстрая разработка программ. Принципы, примеры, практика. – Москва: Вильямс, 2004. – 752 с.
3. Гослинг, Дж., Джой, Б., Стил, Г.Л., Брача, Г., Бакли, А. Язык программирования Java SE 8. Подробное описание. – Москва: Вильямс, 2015. – 672 с.

ПОСТРОЕНИЕ МОДЕЛИ ОДНОЙ ЗАДАЧИ ЛИНЕЙНОГО ПРОГРАММИРОВАНИЯ

Медведева В.Ю.

студентка 4 курса ГрГУ имени Я. Купалы, г. Гродно, Республика Беларусь

Научный руководитель – Сетько Е.А., канд. физ.-мат. наук, доцент

Одним из необходимых условий развития экономической науки является применение точных методов количественного анализа и широкое использование математики. Достижения современной вычислительной техники находят все более широкое применение в экономических исследованиях и планировании. Этому способствует развитие математического программирования, теории игр, теории массового обслуживания.

Задачи линейного программирования (ЗЛП) возникают при разработке программ экономической деятельности, которая всегда происходит при ограничениях, связанных с недостатком ресурсов или директивными заданиями. Они описывают содержательные условия принятия технико-экономических решений. С помощью линейного программирования в достаточно простой и математически строгой форме можно ставить задачи о разработке объемных планов производства, о выборе оптимальных маршрутов транспортных перевозок, о распределении заданий по видам оборудования, о достижении оптимальной загрузки оборудования. Соответствующие расчеты и анализ полученных результатов можно производить на компьютере.

Целью исследования является разработка модели одной задачи линейного программирования с наперед заданными свойствами, что позволяет глубоко изучить все нюансы решения, исследовать её на существование решения, и при положительном ответе на этот вопрос отделить допустимые решения от недопустимых, проанализировать множество допустимых решений и однозначно ответить на вопрос о существовании оптимального решения. Если такое оптимальное решение существует, то найти его.

Материал и методы. Рассмотрим задачу линейного программирования, в которой целевая функция (ЦФ) является линейной функцией, а ограничения задаются с помощью системы линейных неравенств:

$$F(x) = c_1 x_1 + c_2 x_2 \to extr$$

$$\begin{cases} a_{i1} x_1 + a_{i2} x_2 \le b_i, \; i = \overline{1, n_1} \\ a_{k1} x_1 + a_{k2} x_2 \ge b_k, \; i = \overline{1, n_2} \end{cases},$$

где a_{ij}, b_i, c_j – некоторые константы, а $F(x)$ – целевая функция.