МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ

Одеський національний політехнічний університет

Інститут Комп’ютерних Систем

Кафедра Інформаційних Систем

Отчет по практической работе №4

З дисципліни ООП

На тему: «Тесная и слабая связность кода. Отношения между классами. Написание диаграммы классов»

**Виконав студент групи:**

АА-171

Полiкарпов О.В.

**Прийняв:**

Рудніченко Н.Д

Одеса, 2019

**Цель работы**: ознакомиться с различными видами отношений между классами, изучить сценарии использования композиции и наследования, рассмотреть построение диаграммы классов.

**Тесное и слабое связывание классов**

Любое мало-мальски серьезное приложение состоит из двух или нескольких классов, которые взаимодействуют друг с другом, чтобы реализовывать некоторую бизнес-логику. Обычно, каждый объект отвечает за получение ссылок на другие объекты, с которыми он взаимодействует (такие другие объекты называются **зависимостями**, **dependencies**). Такой подход может привести к созданию тесно связанного кода, который тяжело тестировать.

Безусловно, мы не можем избежать связывания вообще, т.к. объектно-ориентированное программирование подразумевает взаимодействие множества объектов различных классов, программа из одного класса не имеет смысла. С другой стороны, нам необходимо избегать **тесного связывания** (tight coupling) классов, так как такой код тяжело повторно использовать, тестировать и тяжело понять, как это всё вместе работает.

В противовес тесному связыванию кода существует принцип **слабо связного** (loose coupling) кода. Слабая связность означает, что изменения, вносимые в один класс, повлекут за собой небольшие изменения в другие классы, что упростит тестирование, рефакторинг, повторное использование кода. Приложение с использованием принципа слабо связного кода легче модифицируется и поддерживается.

**Отношения между классами**

Рассмотрим сходства и отличия между следующими классами: цветы, ромашки, красные розы, желтые розы, лепестки и божьи коровки. Мы можем заметить следующее:

* ромашка – это цветок;
* роза – это (другой) цветок;
* красная и желтая розы – это розы;
* лепесток – это часть обоих видов цветов;
* божьи коровки питаются определенными вредителями, поражающими некоторые цветы.

Из этого простого примера следует, что классы, как и объекты, не существуют изолированно. В конкретной проблемной области, как правило, ключевые абстракции взаимодействуют друг с другом, образуя структуру класса.

Существуют четыре основных типа отношений между классами.

1. Первый тип – **«обобщение/специализация»** (generalization/specialization), или «is a». Например, розы – это цветы, то есть розы являются специализированным подклассом более общего класса «цветы». В Java и многих других ОО-языках программирования, это отношение реализуется с помощью **наследования**.

2. Второй тип – **«целое/часть»** (whole/part), или «part of». Так, лепестки являются частью цветов. Это отношение реализовано в Java с помощью механизмов **агрегации** и **композиции**.

3. Третий тип – **ассоциация**, то есть семантическая зависимость между классами, которые больше никак не связаны друг с другом, например, между божьими коровками и цветами. Другой пример ассоциации представляет собой отношение между розами и свечами, которые можно одновременно использовать для украшения стола.

Ассоциация означает, что объекты двух классов могут ссылаться один на другой, иметь некоторую связь друг с другом. Например, **Менеджер** может выписать **Счет**. Соответственно, возникает ассоциация между **Менеджером** и **Счетом**. Еще пример – **Преподаватель** и **Студент**, то есть какой-то **Студент** учится у какого-то **Преподавателя**. Ассоциация и есть описание связи между двумя объектами.

4. Четвертый тип – **зависимость (dependency)**. Зависимость – отношение между двумя классами, **когда класс А зависит от класса В, но класс В может зависеть, а может и нет, от класса А**. Из этого следует, что изменение в классе B может изменить работу класса А (если же класс B тоже зависит от класса А, то изменение в классе А может изменить работу класса B).

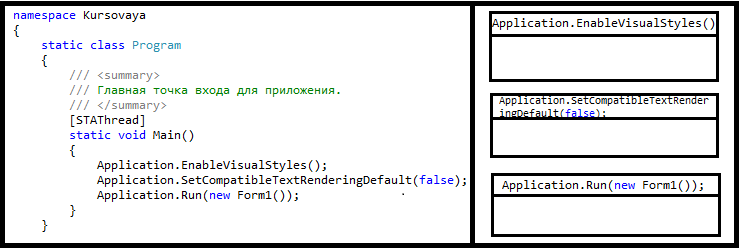


Рис. (1) - главный клас от которого все начинается

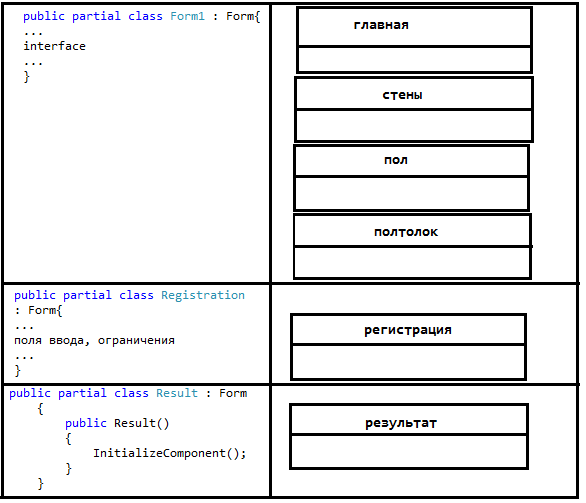


Рис. (2) - классы форм интерфейса, регистрации и результата.

**Вывод**: ознакомился с различными видами отношений между классами, изучил сценарии использования композиции и наследования, рассмотрел построение диаграммы классов.