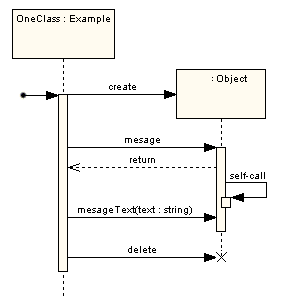
***Практическая работа № 10***

**Построение диаграмм взаимодействия**

Диаграммы взаимодействия – описывают взаимодействие групп объектов в различных условиях их поведения. Наиболее используемым типом данных диаграмм является диаграмма последовательности.

Диаграмма последовательности – это диаграмма, чаще всего, описывающая один сценарий приложения. На диаграмме изображаются экземпляры объектов и сообщения, которыми они обмениваются в рамках одного прецедента (use case).

Пример:



На данной диаграмме, изображены все элементы диаграмм последовательности, за кадром остались только циклы и условия, но о них позже.

На диаграмме последовательности, каждый участник представлен вместе со своей линией жизни (lifeline), это вертикальная линия под объектом, вертикально упорядочивающая сообщения на странице. Важно: все сообщения на диаграмме следует читать сверху вниз. Каждая линия жизни имеет полосу активности (прямоугольники), которая показывает интервал активности каждого участника при взаимодействии.

Участники диаграммы именуются следующим образом: имя : Класс, где и имя, и класс являются не обязательными, но если используется класс, то присутствие двоеточия обязательно.

### Сообщения

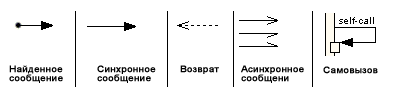
Сообщения можно разделить на 2 вида: синхронные (synchronous message) – требующие возврата ответа и асинхронные (asynchronous message) – ответ не требуется и вызывающий объект может продолжать работу. На диаграмме синхронные вызовы обозначаются закрашенными стрелочками. Асинхронные – незакрашенными или половинными стрелочками.

У первого сообщения, в нашей диаграмме, нет участника, пославшего его, поскольку оно приходит от неизвестного источника. Такое сообщение называется найденным сообщением (found message).

Обратной пунктирной стрелкой показан возврат ответа на сообщение message (т.к. сообщение message является синхронным). Лучше применять изображение возврата только в тех случаях, когда это поможет лучше понять устройство взаимодействия. Во всех остальных случаях, стоит опускать изображения возвратов, т.к. они будут вносить некоторую неразбериху. Просто, при использовании синхронного сообщения, стоит помнить, что у него всегда есть возврат.

Если в сообщении требуется передать параметры, то они указываются в скобках через запятую, с указанием типа параметра (см. сообшение messageText(text : string) ).

Подведем итог знаний о сообщениях в виде небольшой картинки-памятки:



### Создание и удаление участников

Тут вообще все просто. На первом рисунке создание нового объекта – это сообщение create. Создаваемый объект находится всегда ниже объекта, из которого он был создан. Если при создании применяется конструктор то имя сообщения не обязательно, но для ясности все же лучше писать new.

Удаление участника обозначается крестом. Стрелка сообщения, идущая к такому кресту, означает, что один участник, явным образом удаляет другого участника. Если же крестик стоит в конце линии жизни участника, и к нему не идет никаких сообщений – это означает, что участник удаляет сам себя, т.е. срабатывает деструктор.

### Циклы, условия и другие

Диаграммы последовательностей хороши, когда требуется графически представить взаимодействие нескольких объектов, в рамках определенного сценария. Но вот для точного определения поведения объектов, эти диаграммы подходят гораздо хуже.

Разработчики UML все же нашли способ отображать на диаграмме циклы и условные операции. Для этого используются фреймы взаимодействий (interaction frames). Однако, фреймы взаимодействий очень сильно “захламляют” диаграмму и иногда она становится практически не читаемой.

Фреймы взаимодействия, это нововведение UML 2, но многие разработчики, при обозначении циклов и условий, продолжают пользоваться приемами из UML 1. В UML 1, для этих целей использовались маркеры итераций и маркеры защиты.

Маркер итераций (iteration marker ) обозначается символом \*, добавленным к имении сообщения. Для уточнения количества итераций, к маркеру можно добавить текст в квадратных скобках, например, маркер – \* [для всех элементов массива], показывает, что сообщение, к которому он добавлен, должно быть применено ко всем элементам массива.

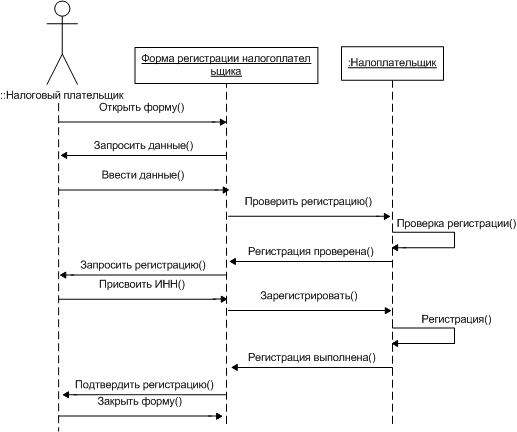
Защита (guard) – это условное выражение, которое записывается в квадратных скобках и означающее, что сообщение должно быть послано только когда защита принимает истинное значение.

UML можно и нужно применять в повседневных разработках.

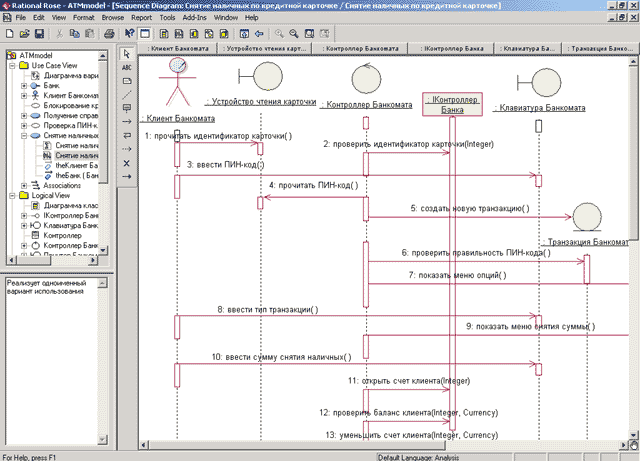
***Задание***

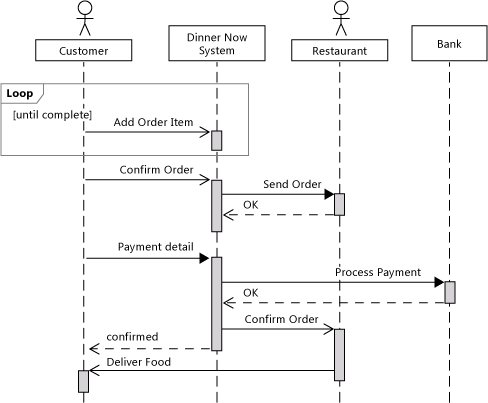
1. Составить **диаграмму последовательности** для Вашей задачи, она отражает упорядочение по времени сообщений, передаваемых между объектами

Образец составления представлен примерами ниже.



Пример 1

Пример 2



Пример 3