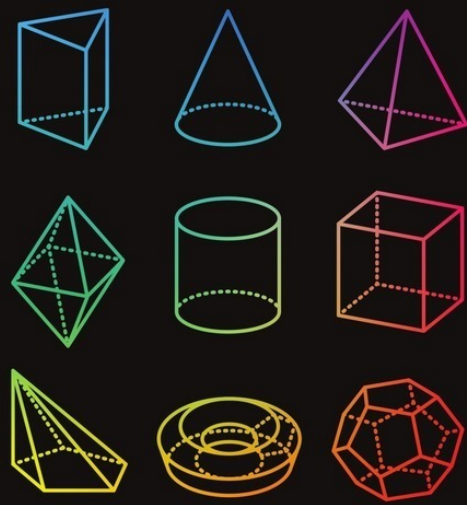
ПРОЕКТИРОВАНИЕ НА UML

Сборник задач



Хританков • Полежаев • Андрианов

Антон Сергеевич Хританков
Андрей Иванович Андрианов
Валентин Александрович Полежаев

Проектирование на UML. Сборник задач

http://www.litres.ru/pages/biblio_book/?art=27097814
ISBN 9785448579547

Аннотация

В данном сборнике представлены задачи по проектированию ПО с использованием унифицированного языка моделирования UML 2, принципов и паттернов проектирования. Сборник содержит более 120 задач с несколькими заданиями в каждой по разным разделам UML и проектирования ПО. Для каждого раздела приводятся основные понятия, для задач даны ответы и пояснения по решению. http://www.objectoriented.ru

Содержание

Проектирование на UML. 5
ОБ АВТОРАХ 7
ПРЕДИСЛОВИЕ КО ВТОРОМУ ИЗДАНИЮ 9
ПРЕДИСЛОВИЕ К ПЕРВОМУ ИЗДАНИЮ 12
ГЛАВА 1. ОСНОВЫ ОБЪЕКТНО-ОРИЕНТИРОВАННОГО МОДЕЛИРОВАНИЯ 14
§1. КЛАССЫ И ОБЪЕКТЫ 18
ОСНОВНЫЕ ПОНЯТИЯ 18
ЗАДАЧИ 22
§2. СЦЕНАРИИ И ВАРИАНТЫ ИСПОЛЬЗОВАНИЯ 30
ОСНОВНЫЕ ПОНЯТИЯ 30
Конец ознакомительного фрагмента. 32

Проектирование на UML
Сборник задач

Антон Сергеевич Хританков
Валентин Александрович Полежаев
Андрей Иванович Андрианов

ISBN 978-5-4485-7954-7
Создано в интеллектуальной издательской системе Ridero

Проектирование на UML. Сборник задач по проектированию программных систем

Рекомендовано ученым советом ФИВТ МФТИ
к использованию в учебном процессе факультета
при подготовке студентов по направлениям
010400 «Прикладные математика и информатика» и
010600 «Прикладные математика и физика»
Рецензенты:
д.ф.-м.н., профессор, Соколинский Л. Б.,
ведущий разработчик, Колпаков Е. А.

Аннотация

В данном сборнике представлены задачи по проектированию программных систем с использованием унифицированного языка моделирования UML2, принципов и паттернов проектирования. Сборник содержит более 120 задач с несколькими заданиями в каждой по разным разделам UML и проектирования ПО. Для каждого раздела приводятся основные понятия, для задач даны ответы и пояснения по решению. Приведены рекомендации по составлению проверочных работ с использованием задач сборника по темам

проектирования.

Для слушателей курсов по объектно-ориентированному анализу и проектированию программного обеспечения, студентов технических и физико-математических специальностей, преподавателей высших учебных заведений, специалистов по программной инженерии.

Дополнительную информацию и материалы можно найти на сайте книги http://www.objectoriented.ru

При цитировании, используйте следующую информацию о книге.

Хританков А. С., Полежаев В. А., Андрианов А. И.

Проектирование на UML. Сборник задач по проектированию программных систем. 2-е. изд. – Екатеринбург.: Издательские решения, 2017. – 240 с.; ил.

ISBN 978-5-4485-7954-7

УДК 004.41+004.02+372.8

ББК 32.973.23—018

ОБ АВТОРАХ

Хританков Антон Сергеевич, к.ф.-м.н.

доцент кафедры АТП, Московский физико-технический институт.

Защитил диссертацию в сфере высокопроизводительных вычислений (МФТИ / ИСА РАН). Опыт преподавания более восьми лет, научные интересы: архитектура программного обеспечения, автоматизированные и интеллектуальные методы разработки программ. Опыт работы в индустрии более 12 лет от разработчика ПО до архитектора и руководителя департамента разработки и исследований. Сертифицированный специалист по UML2 (OMG Certified UML Professional Advanced).

Email: anton.khritankov@objectoriented.ru

Полежаев Валентин Александрович

директор по разработке и анализу данных компании Интелиор.

Окончил ВМК МГУ, автор нескольких статей по теме машинного обучения и практике применения предметно-ориентированных методов проектирования. Участвовал в разработке более десяти информационных систем, из них более половины в качестве бизнес-аналитика и архитектора.

Email: valentin.polezhaev@objectoriented.ru

Андрианов Андрей Иванович

руководитель группы морфологии «Аби Продакшн» (ABBYY).

Магистр физ.-мат. наук (МФТИ), в разное время преподавал в МФТИ курсы «Алгоритмы и структуры данных», «Проектирование программных систем», «Машинное обучение». А также «Концепции языков программирования», «Промышленное программирование». Опыт разработки, проектирования архитектуры и управления проектами более 9 лет.

Email: andrey.andrianov@objectoriented.ru

ПРЕДИСЛОВИЕ КО ВТОРОМУ ИЗДАНИЮ

Проектирование – это процесс построения модели объекта, который предполагается разработать или создать. Модели в проектировании играют важную роль: модели используются для уточнения того, что нужно сделать, для прояснения способа реализации, для понимания реализуемости решения и его соответствия требованиям, для передачи знаний и распространения информации о проектируемом объекте, для планирования и ведения работ по реализации.

Во многих инженерных отраслях проектирование занимает важное место и часто регулируется государственными, промышленными стандартами или стандартами уровня предприятия. В сфере разработки программного обеспечение проектирование будущей программной системы происходит как в начале работ, так и по ходу реализации.

Важно понимать, что область знаний проектирования – это отдельная дисциплина, требующая особых навыков работы с моделями, применения методов и практик, которые обычно не используются при реализации создаваемого объекта.

На данный момент в индустрии разработки программного обеспечения сложилось несколько отраслей, каждая из которых использует несколько отличные от других методы проек-

тирования. Среди этого разнообразия в книге уделено больше внимания проектированию прикладных программ, компонентов и приложений с помощью объектно-ориентированных методов и унифицированного языка моделирования UML2.

Овладение этими методами позволит в дальнейшем с легкостью освоить и другие сферы проектирования программных систем, и другие языки моделирования.

В данной книге собрано более сотни задач по проектированию. Авторы приложили все возможные усилия к тому, чтобы задачи помогли читателю понять смысл и освоить те или иные концепции, принципы и методы проектирования. Рассматриваемый перечень тем примерно соответствует программе курса по проектированию программных систем, читаемом авторами в Московском физико-техническом институте на протяжении уже более восьми лет и рекомендациям ACM/IEEE по составу учебных программ по проектированию программного обеспечения.

По сравнению с первым изданием книгу пополнили задачи, предлагавшиеся студентам на контрольных работах по проектированию программного обеспечения, вошел новый раздел по предметно-ориентированному проектированию, добавлены рекомендации по составлению проверочных и контрольных работ по отдельным темам проектирования, а также вошло множество задач разной сложности, которые авторы сочли интересными и важными для освоения дисци-

плины проектирования.

По сравнению с первым изданием в книге изменился состав авторов, тем не менее, важно отметить вклад Штукатурова А.Н, по согласованию с которым во второе издание вошли подготовленные им задачи 2.5, 3.7, 3.9, 4.4, 5.5, 5.6, 7.5, 8.9. Раздел §9 подготовлен Полежаевым В. А., задачи 6.2, 6.4, 2.3, 2.8, 3.10, 7.11, 7.12, 1.3, 1.4, 2.1, 7.1, 8.1, 3.2 предложены Андриановым А. И., остальные задачи, теоретическая справка и примеры решения задач составлены Хританковым А. С.

Апрель 2017

ПРЕДИСЛОВИЕ К ПЕРВОМУ ИЗДАНИЮ

Предлагаемая читателю книга является сборником задач по курсу проектирования программных систем, преподаваемому авторами в Московском физико-техническом институте.

Сборник включает задачи по проектированию и моделированию с помощью языка UML2. Каждая задача имеет условие, в котором описана заготовка модели, и несколько вопросов к ней. Часть вопросов направлена на уточнение и расширение заготовки модели. Другая часть проверяет понимание смысла построенной модели. Предполагается, что вопросы к задаче будут решаться по порядку.

Задачи сборника направлены на развитие навыков изложения проектных решений средствами UML и устроены таким образом, чтобы в наибольшей степени обеспечить единственность решения. В сложных случаях к задачам даны пояснения и указания по решению.

В сборник включены задачи по объектно-ориентированному моделированию предметной области, в том числе задачи на выделение классов, задачи по моделированию структур времени выполнения и размещения компонентов программных систем. Моделированию поведения систем в сборнике посвящены разделы описания взаимодействий, моде-

лирования с помощью конечных автоматов и представления деятельности.

Разделы сборника снабжены пояснениями по основным понятиям. В конце сборника приведены ответы к задачам с пояснениями, приводятся примеры решения некоторых задач.

Сборник может быть использован при проведении семинаров по курсам объектно-ориентированного моделирования и программирования, проектирования программных систем, а также специализированных курсов по языку UML. Задачи сборника могут предлагаться в качестве примеров, демонстрирующих и поясняющих основные понятия и особенности языка, могут входить в задания для самостоятельной работы, проверочные и контрольные работы, использоваться для самостоятельного изучения методов проектирования и языка UML.

Задачи составлены авторами на основе собственного опыта преподавания, адаптированы из профессиональной практики, проведенных контрольных и проверочных работ, предлагаемых студентам проектов.

Июнь 2012

ГЛАВА 1. ОСНОВЫ ОБЪЕКТНО-ОРИЕНТИРОВАННОГО МОДЕЛИРОВАНИЯ

Краткая история UML. Унифицированный язык моделирования UML появился в результате объединения нескольких подходов к моделированию в середине 1990-х годов. В отличие от предыдущих попыток, в создании языка участвовали авторы этих подходов, а вследствие стандартизации через организацию OMG, участвовали также заинтересованные компании и исследовательские коллективы из разных отраслей.

Поэтому разные части UML отражают потребности в моделировании в разных отраслях и объединение их в одном языке и на основе одной базовой системы понятий позволяет говорить как раз об унифицированном языке моделирования.

В данной книге унифицированный язык моделирования выбран как основной для выражения проектировочных решений. При решении задач стоит ориентироваться на версию UML 2.4.1, которая была стандартизирована международной организацией по стандартизации ISO как ISO/IEC 19505—1:2012 и ISO/IEC 19505—2:2012.

Уровни использования UML. Выделяют несколько

уровней владения и использования UML и моделей в целом при проектировании программных систем. На уровне эскиза модели используются для пояснения решений, неформального общения, документирования и не обладают полнотой, строгостью и могут быть несогласованными. На уровне спецификации модель используется как чертеж или план реализации, в соответствии с которым разрабатывается программная система.

Модели и диаграммы на этом уровне должны следовать нотации языка и быть **согласованными** (**well-formed**). Согласованность модели означает соответствие правилам использования языка UML2, определенным в его спецификации (метамодели) [4]. Например, если на диаграмме показан элемент модели, то в модели также должны быть определены все элементы, используемые показанным.

На исполняемом уровне модель представляет собой достаточное описание системы для ее воплощения автоматическими средствами. В этом случае исходный код системы может не сохраняться вовсе и быть промежуточным этапом получения работающей программной системы из исходных моделей.

В данном сборнике задач следует ориентироваться на использование UML на уровне спецификации. В то же время часть задач предполагает владение языком на исполняемом уровне.

Решение задач. Прежде чем приступить к решению за-

дач стоит ознакомиться с рекомендуемой литературой для ознакомления с методами проектирования и нотацией. В помощь читателю в начале каждого раздела приводится краткая справка по используемым в задачах раздела понятиям и демонстрируется нотация языка.

Все задачи построены по единому принципу. В условии дается заготовка модели. Это может быть диаграмма или текстовое описание. В текстовом описании названия элементов модели приведены *курсивом* для облегчения их восприятия. Далее приводится несколько заданий или вопросов к условию. В качестве решения задания нужно указать по шагам ход рассуждения от условия или предыдущего задания к достижению условий, указанных в задании. Для ответа на вопрос следует привести рассуждение в обоснование полученного ответа и сам ответ. Задания и вопросы к задачам следует выполнять по порядку. Решение следующего задания может зависеть от решения предыдущего. Ответом на задание будет фрагмент диаграммы или нескольких диаграмм с представлением изменений, требуемых в данном задании.

При решении следует руководствоваться условием задачи, знаниями методов решения, нотацией и значением понятий языка моделирования. Часть задач составлена на основе реальных проектов разработки программного обеспечения, другую часть составляю учебные задачи. Такие задачи могут вызывать ассоциации с похожими ситуациями или реальными объектами. В этом случае следует придерживаться усло-

вия задачи. Если не указано иное, решение задач не предполагает каких-либо специальных знаний в специализированных областях. При необходимости дается сноска, где можно получить дополнительную информацию.

Задачи и задания повышенной сложности отмечены звездочкой (*), для некоторых задач приведено решение, в этом случае указана страница, на которой оно расположено (см. решение в §11). Перед тем, как приступить к решению задач рекомендуется ознакомиться с примерами решения и понять порядок ведения рассуждения и степень его детальности.

При составлении задач уделялось особое внимание тому, чтобы решение было единственным. При необходимости в заданиях к задачам даются указания по предполагаемому способу решения. Впрочем, вполне возможно, что читатель сможет предложить более удачные или лаконичные решения по некоторым задачам. Возможность существования лучшего решения следует учитывать и не требовать однозначного совпадения решения с ответами, приводимыми авторами сборника.

§1. КЛАССЫ И ОБЪЕКТЫ

ОСНОВНЫЕ ПОНЯТИЯ

Пространство имен (namespace) – это именованный элемент модели, который может содержать другие именованные элементы. Принадлежность пространству имен показывается отношением **включения в пространство имен (membership)**. **Полностью квалифицированное имя (fully-qualified name)** элемента в модели состоит из последовательности имен всех вложенных пространств имен, в которые включен элемент.

Классификатор (classifier) – это пространство имен в модели, указывает на общие некоторому множеству объектов черты. Черты классификатора могут быть поведенческими, структурными или соединительными.

Класс (class) – это классификатор, который описывает некоторую концепцию моделируемой области. Черты класса могут быть различных видов, наиболее часто для описания функциональности класса используются операции (operation), а для описания хранимых данных или связей с другими классами – **свойства (property)**. Если типом свойства является примитивный тип или тип данных, свойства показывают как атрибуты, класса иначе как часть ассо-

циации.

Операция (operation) – черта поведения интерфейса, класса или типа данных. Операция задается именем, набором параметров, типом возвращаемого значения и его кратностью. Каждый параметр операции может иметь имя, тип, кратность. В программировании операции будет соответствовать сигнатура метода.

Обратите внимание, что определение операции в классе не влечет определение ее реализации в этом классе. Понятие метода в UML2 обозначает реализацию операции алгоритмом, который не описывается средствами UML или не уточняется в модели. В последнем случае, такую реализацию операции называют нечетким поведением (opaqueBehavior).

Интерфейсом (interface) называют особый вид классификатора, который определяет способ взаимодействия с экземпляром класса, реализующего интерфейс. Интерфейс обычно включает операции, но может включать и свойства. В последнем случае наличие указанных свойств является обязательным для реализующего интерфейс класса.

Экземпляр класса (instance) – это элемент модели с описанием, возможно неполным, объекта, которому в системе приписаны черты данного класса. Для того чтобы указать значения свойствам класса в экземпляре используют слоты.

Связью (link) называется экземпляр ассоциации, соединяющий экземпляры классов. В языке программирова-

ния однонаправленной связи соответствует типизированный указатель или ссылка.

Ассоциация (association) – это типизированное отношение между классами, которое указывает на логическую связь между ними. Ассоциация имеет два или более полюсов, по одному у каждого связанного класса. Название полюса обычно указывает на роль, которую класс играет в ассоциации.

Обобщение (generalization) является направленным отношением от более специализированного классификатора к более общему. Специализированный, или дочерний, классификатор наследует черты более общего, или родительского, классификатора. Отношение обобщения уточняется отдельно для каждого вида классификатора, в том числе для классов и интерфейсов.

Украшениями (adornments) называются свойства полюса ассоциации, уточняющие роль участвующего в ассоциации класса. С помощью украшений указываются направление навигации, вид композиции и другие свойства полюса.

Типом данных (data type) называется классификатор, экземпляры которого не обладают индивидуальностью и, при совпадении значений свойств, взаимозаменяемы. **Простыми (primitive)**, или примитивным типами данных, являются предопределенные типы: целое *Integer*, строка *String*, логический тип *Boolean*, числа с плавающей запятой *Real* и неограниченные натуральные числа *UnlimitedNatural*, ко-

торые используются для моделирования неопределенного количества элементов, например, экземпляров класса, участвующих в ассоциации.

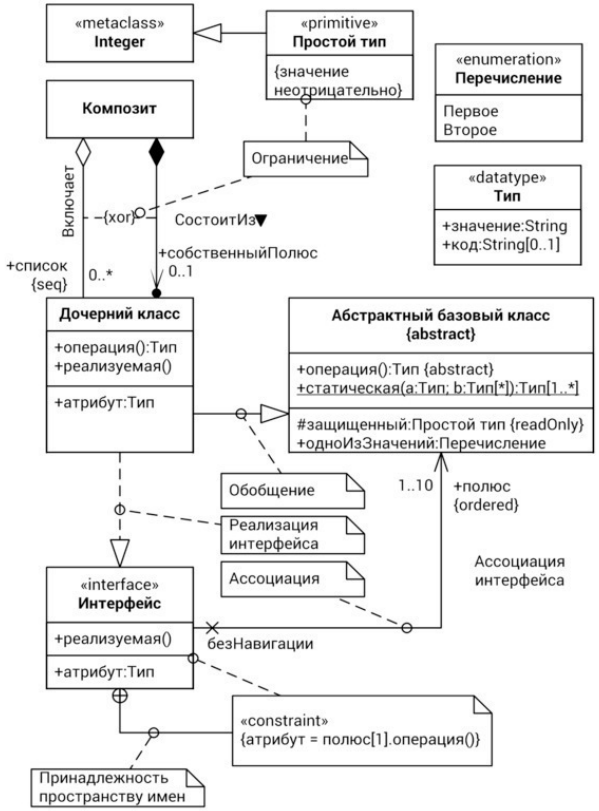


Рис. 1. Основная нотация диаграмм классов

Рис. 2. Нотация диаграмм экземпляров

Ограничением (constraint) называется логическое выражение об ограничиваемых элементах модели, вычисляемое в контексте какого-либо элемента. Если выражение ложно, то модель считается противоречивой (ill-formed).

Примеры нотации указанных выше элементов модели приведены на рис. 1 и рис. 2.

ЗАДАЧИ

1.1. Абстрактный класс *Account* имеет два дочерних класса: счет физического лица *PersonalAccount* и юридического *CompanyAccount*. При решении задачи используйте диаграм-

мы классов.

а. Добавьте класс *Person* с общедоступным атрибутом *FullName* строкового типа и свяжите его с классом *PersonalAccount* ассоциацией *Owns* с полюсом *owner* у *Person* и навигируемым полюсом *account* у *PersonalAccount*.

б. Аналогично для счета юридического лица добавьте владельца *Company*, свяжите анонимной ассоциацией с *CompanyAccount* и укажите подходящие названия полюсов.

в. Добавьте класс адреса *Address* с атрибутами строкового типа *street*, *city* и целочисленным положительным *building*. Укажите с помощью новых анонимных ассоциаций, что *Person* может иметь адрес регистрации *registeredAt*, фактический адрес *actual*, в то время как компания связана с одним юридическим адресом *legalAddress* и может иметь почтовый адрес *postAddress*.

1.2. Интерфейс *Stack* определяет операции помещения в стек *push* с параметром *obj* типа *Element*, операцию получения элемента из стека *pop* с возвращаемым значением типа *Element*. При решении задачи используйте диаграммы классов.

а. Добавьте в интерфейс *Stack* операции очистки стека *reset*, которая не имеет параметров, статическую операцию создания нового стека *createNew* с возвращаемым значением типа *Stack*.

б. Покажите, что интерфейс *Stack* зависит от типа данных *Element*.

в. Добавьте класс *ListStack*, который реализует интерфейс *Stack*. Покажите реализуемые классом операции интерфейса.

г. Добавьте в класс *ListStack* частное структурное свойство *arr* типа *Element* с кратностью больше нуля, значения которого упорядочены и могут повторяться.

д. Добавьте частный целочисленный атрибут *increment* только для чтения и защищенную операцию изменения размера *resize* с целочисленным параметром *newSize*.

е. Покажите на диаграмме экземпляров экземпляр *stack* класса *ListStack*, свойство *arr* которого содержит элемент *first* типа *Element* первым и *second* того же типа вторым. Укажите, что атрибут *increment* экземпляра *stack* равен *10*.

1.3. В пространстве имен *Time* расположены перечисления *Month*, *DayOfWeek*, а также классы *Date* и *Period*. При решении задачи используйте диаграммы классов.

а. Укажите, что перечисление *Month* может принимать значения: *Jan*, *Feb*, *Mar*, *Apr*, *May*, *Jun*, *Jul*, *Aug*, *Sep*, *Oct*, *Nov*, *Dec*.

б. Укажите, что перечисление *DayOfWeek* может принимать значения: *Mon*, *Tue*, *Wed*, *Thu*, *Fri*, *Sat*, *Sun*.

в. Добавьте классу *Date* частные атрибуты *year*, *month*, *dayOfMonth* типа *Integer*, а также общедоступные операции:

– получения года *getYear* типа *Integer*; – получения месяца *getMonth* типа *Month*; – получения дня *getDayOfMonth* типа *Integer*; – получения дня недели *getDayOfWeek* типа

DayOfWeek.

г. Добавьте классу *Date* общедоступную статическую операцию *now ()* типа *Date*.

д. Добавьте классу *Period* общедоступную статическую операцию *between*. У операции два аргумента: *from* и *to*. Оба аргумента имеют тип *Date*. Операция возвращает значение типа *Period*

е. Добавьте классу *Date* операцию *plus* с аргументом *delta* типа *Period*. Результат операции – значение типа *Date*.

1.4. Класс *MyWindow* уточняет абстрактный базовый класс *Window*. *MyWindow* состоит (композиция) из кнопки класса *Button* и надписи класса *Label*. Отобразите на диаграмме классов.

а. Класс *Label* имеет частный атрибут *text* типа *String* и общедоступную операцию *setText* с параметром *text* типа *String*.

б. Композиция между *MyWindow* и *Button* называется *HoldsButton*. Полюс со стороны кнопки имеет имя *okButton*, защищенную видимость, кратность *1*. Композиция между *MyWindow* и *Label* называется *HoldsLabel*. Украшения полюса со стороны *Label*: название *textLabel*, частная видимость, кратность *1*.

в. Для реакции на события кнопки реализован паттерн Слушатель (Listener) следующим образом. Класс *Button* предоставляет операцию *setClickListener* с единственным параметром *l* типа *IClickListener*. Интерфейс *IClickListener* содержит единственную операцию *onClick* без параметров.

г. Класс *MyWindow* реализует интерфейс *IClickListener* для реакции на нажатие кнопки. Отобразите на диаграмме, что между классом *Button* и *MyWindow* есть ассоциация с именем *NotifyListener* с направлением от кнопки к окну. Укажите, что полюс со стороны окна называется *listener*, имеет тип *IClickListener*, множественную кратность и частную видимость.

д. И *Label* и *Button* имеют строковый атрибут *text*, который можно менять с помощью метода *setText*. Вынесите общий атрибут и метод в абстрактный базовый класс *TextWidget*.

е. Отобразите на диаграмме объектов, как в процессе выполнения объекты связаны между собой: объект *window* класса *MyWindow* связан с кнопкой *button* класса *Button* и с надписью *label* класса *Label*.

1.5. (см. решение в §11) Интерфейс доступа к коллекции элементов *Collection* обобщает интерфейс работы со списками *List*. Абстрактный класс *BaseCollection* реализует интерфейс *Collection*, абстрактный класс *BaseList* является потомком *BaseCollection* и реализует интерфейс *List*, оставляя операции по хранению данных дочерним классам.

а. Используя наследование, добавьте в модель класс *ArrayList*, реализующий операции со списками с помощью массива.

б. Пусть интерфейс *List* содержит операцию *get* получения элемента списка по заданной позиции *k*. Укажите, в каких классах должна быть объявлена данная операция, чтобы мо-

дель была согласованной. Ответ поясните.

в. Пусть интерфейс *Collection* содержит операцию *add* добавления элемента *obj*. Укажите, в пространстве имен каких классов может присутствовать поведение, реализующее операцию *add*. Ответ поясните.

1.6. Класс *Collections* содержит общедоступную статическую операцию *addAll* с возвращаемым значением типа *boolean*. Первый параметр операции называется *coll* и имеет тип *Collection*, второй параметр называется *elements* и имеет тип *Object* и кратность больше нуля.

а. Добавьте в класс *Collections* статический атрибут *empty* типа *Collection,* предназначенный только для чтения.

б. Реализуйте в классе *Collections* операцию *addAll* с помощью нечеткого поведения (метода), используя операцию добавления элемента *insert (e: Object)* класса *Collection*. *Указание*. Алгоритм реализации можно показать как псевдокод в комментарии в формате *{method = {<language>} <method body>}*.

1.7. Узел дерева *Node* может иметь несколько дочерних *child* узлов того же класса *Node*.

а. Приведите пример бинарного дерева, состоящего из семи узлов *Node*.

б. Постройте модель дерева, в котором каждый узел имеет от двух до четырех дочерних узлов.

в. Разработайте модель дерева, узлы которого могут быть двух видов: узел *Red* и узел *Black*. *Указание.* Вид узла может

изменяться, при этом следует считать, что поведение узла не изменяется при смене типа.

1.8. У абстрактного класса заказа *Reservation* имеется два потомка: одиночный *Single* и подписка *Subscription*. *Single* связан с одним билетом *Ticket* ассоциацией бронирован *reserved*, *Ticket* может быть связан той же ассоциацией не более чем с одним *Single*.

а. Свяжите подписку с билетами в количестве от трех до шести включительно. Билет не обязательно связан с подпиской.

б. Как с помощью ограничений указать, что билет не может быть одновременно связан и с подпиской, и с одиночным заказом?

в. Пусть одиночная подписка наследует свойства одиночного заказа и подписки. С каким максимальным количеством билетов она может быть связана? Ответ поясните.

1.9. Умный дачный домик *SmartHouse* состоит из четырех стен *Wall* и крыши *Roof*. Домик реагирует на штормовые предупреждения *stormWarning* и укрепляет крышу *harden*, закрывает окна *closeWindows* в стенах. Используемые стройматериалы *Material* характеризуются ценой *price* и удельным весом *unitWeight*.

а. Добавьте стройматериалы для постройки домика: красный и белый кирпич *Brick*, доски *Plank* из сосны и дуба.

б. Укажите, что кирпич является материалом *material* стен. Используя ассоциации, покажите, что каркас крыши

Frame сделан из не более чем сорока досок и может быть одного из видов *FrameKind*: мансарда, плоский или треугольный.

в. Каркас можно покрыть стройматериалом черепица *Tiling*, отразите это в модели.

г. Допустим, изобретен универсальный стройматериал, заменяющий доски, кирпичи и черепицу. Постройте из него дачный домик. Сколько экземпляров материала понадобится? Ответ поясните.

§2. СЦЕНАРИИ И ВАРИАНТЫ ИСПОЛЬЗОВАНИЯ

ОСНОВНЫЕ ПОНЯТИЯ

Актором (actor) называется классификатор, который моделирует пользователя или систему, внешнего по отношению к моделируемой системе или компоненту. Акторов, которые используют систему для достижения собственных целей, называют основными. Акторов, которых система использует для достижения целей других акторов, называют второстепенными.

Вариантом использования (use case) называют классификатор, который описывает совокупность сценариев взаимодействия акторов с системой или компонентом для достижения какой-либо цели, значимой для акторов. Варианты использования могут различаться по уровню цели, достижение которой они обеспечивают: высокоуровневые цели, пользовательские цели и отдельные функции системы.

Субъектом (subject) варианта использования называют систему или компонент, взаимодействие акторов с которым он описывает.

Ассоциация (association) актора с вариантом использования указывает на взаимодействие актора с субъектом в од-

ном из сценариев данного варианта использования.

Отношение **расширения (extension)** между вариантами использования указывает, что при выполнении заданного в **точке расширения (extension point)**

Конец ознакомительного фрагмента.