

Лабораторная работа № 10

НАСЛЕДОВАНИЕ, ПОЛИМОРФИЗМ

1. Наследование – тип связи между классами, когда один из классов, называемый родительским или базовым содержит функциональное наполнение (свойства и методы), используемое (наследуемое) другими классами, называемыми дочерними. Дочерний класс может содержать не только «скопированные» (унаследованные) из базового класса функциональные блоки, но и иметь собственные, специфичные только для него свойства и методы.

Наследование возникает тогда, когда два типа объектов содержат схожие характеристики и поведение, но один тип содержит больше характеристик, чем другой, и может выполнять больше действий или выполнять их по другому.

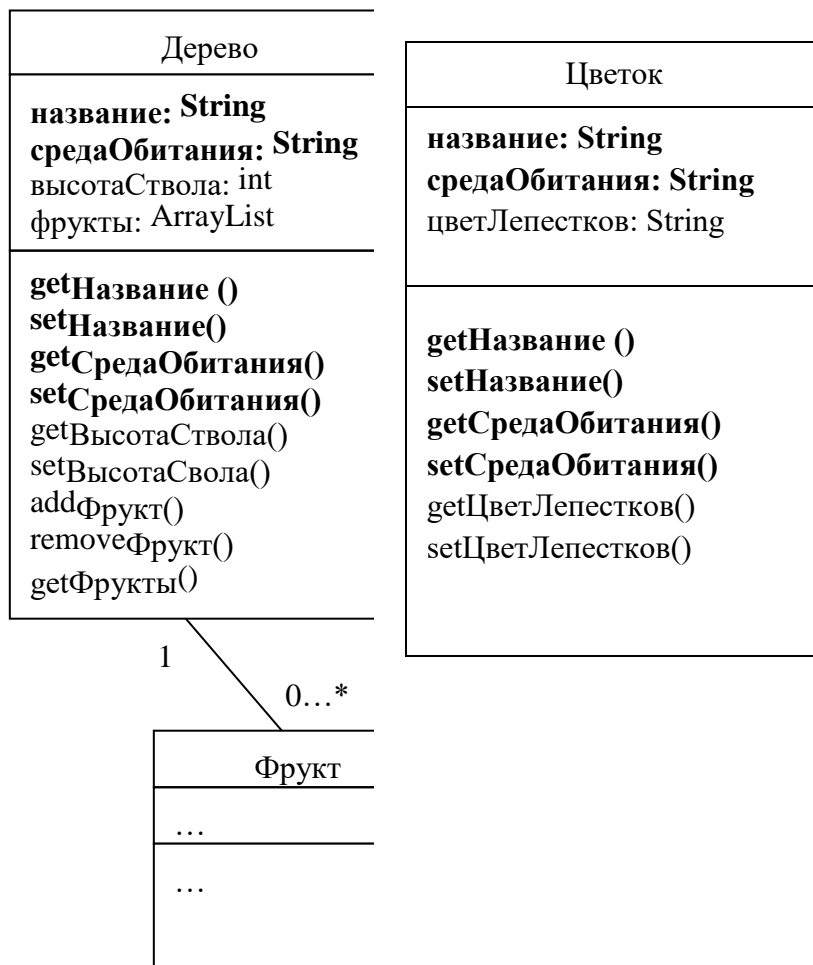
Например, требуется учитывать и обрабатывать два типа объектов: Дерево и Цветок. Дерево обладает следующим набором свойств:

- название
- среда обитания
- высота ствола
- фрукты, произрастающие на дереве

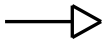
Тип Цветок обладает следующим набором свойств:

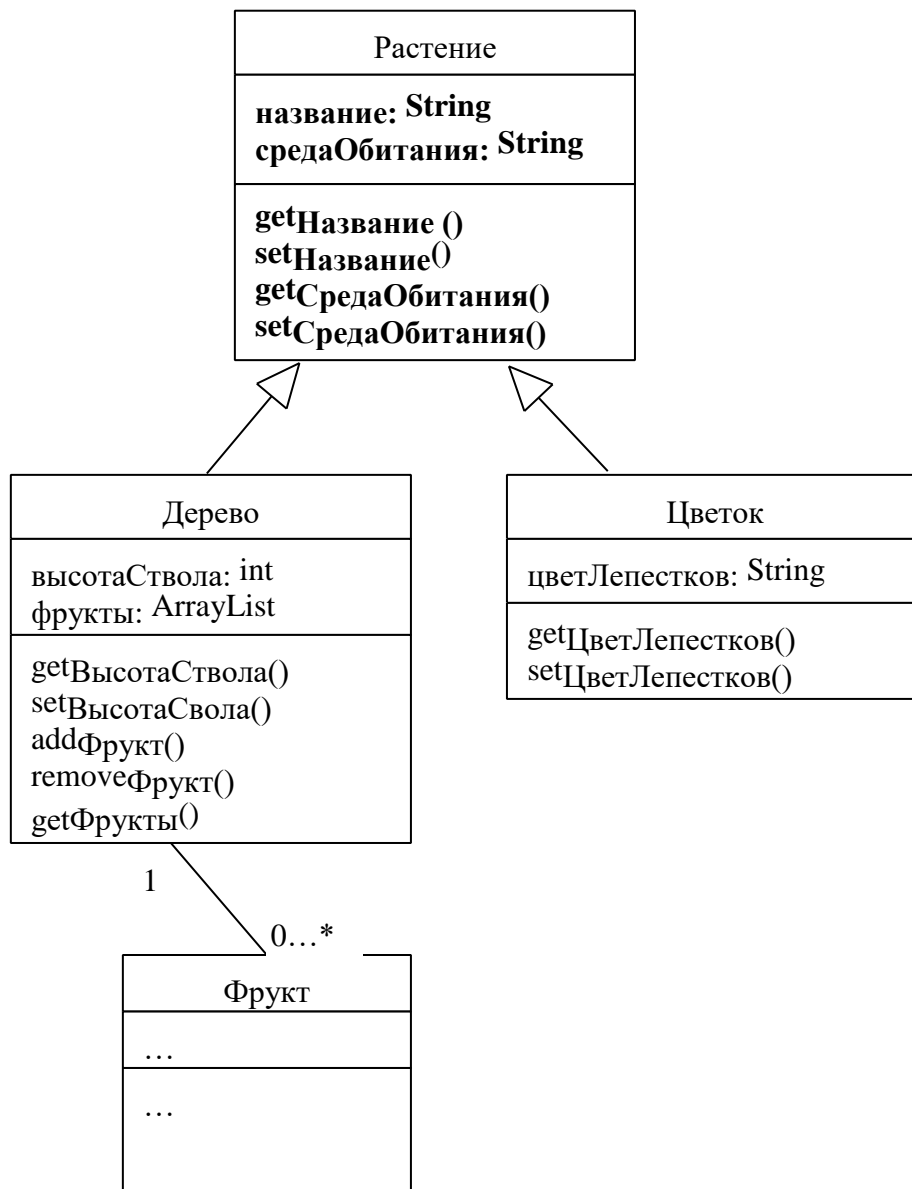
- название
- среда обитания
- цвет лепестков

В ООП указанные типы могут быть представлены в виде двух классов (параметры и типы возвращаемых значений методов не указаны в целях экономии места):



Как видно на приведенной UML-диаграмме, в обоих классах свойства Название и средаОбитания полностью дублируют. Дублируются также методы getНазвание(), setНазвание(), getСредаОбитания(), setСредаОбитания(). Применяя концепцию наследования, можно вынести дублируемые свойства и методы

в базовый класс Растение, а классы Дерево и Цветок унаследовать от класса Растение. Связь типа «Наследование» в UML изображается стрелкой вида: 



В JAVA наследование реализуется с помощью ключевого слова `extends` <базовый класс>, указываемого в заголовке дочернего класса, например:

```
class Plant {
    ...
}

class Tree extends Plant {
    ...
}

class Flower extends Plant {
    ...
}
```

2. Абстрактные классы

Класс является абстрактным в том случае, если не может существовать объектов данного класса. Смысл объявления абстрактного класса заключается в задании базового набора свойств и методов, наследуемых дочерними классами.

Наследование – это связь типа «является». Абстрактный тип объектов возникает тогда, когда в рамках заданной предметной области имеет место полная классификация объектов данного типа по подтипам – т.е. любой объект, принадлежащий базовому типу, может быть отнесен к одному из дочерних типов.

Например, в саду растут **только** Деревья и Цветы. Таким образом, любое Растение в саду может быть отнесено либо к классу Деревьев, либо к классу Цветов, т.е. имеет место полная классификация. В результате класс Растение можно представить в виде абстрактного класса.

В JAVA абстрактный класс объявляется с помощью ключевого слова `abstract`, например:

```
abstract class Plant {  
    ...  
}
```

Так как у абстрактного класса не может быть объектов, то:

```
new Plant(); // является недопустимой операцией
```

Тем не менее, сохраняется возможность создания ссылок абстрактного типа, например:

```
Plant p; // является допустимой операцией
```

3. Переопределение методов

Предположим, требуется формировать и выводить на экран описание растений (т.е. деревьев и цветов), произрастающих в саду. При этом описание для деревьев и описание для цветов формируется различным образом (в <...> подставляются значения свойств):

Для дерева:

«Название дерева: <название>, Среда обитания: <среда обитания>

Высота ствола: <высота ствола> м.,

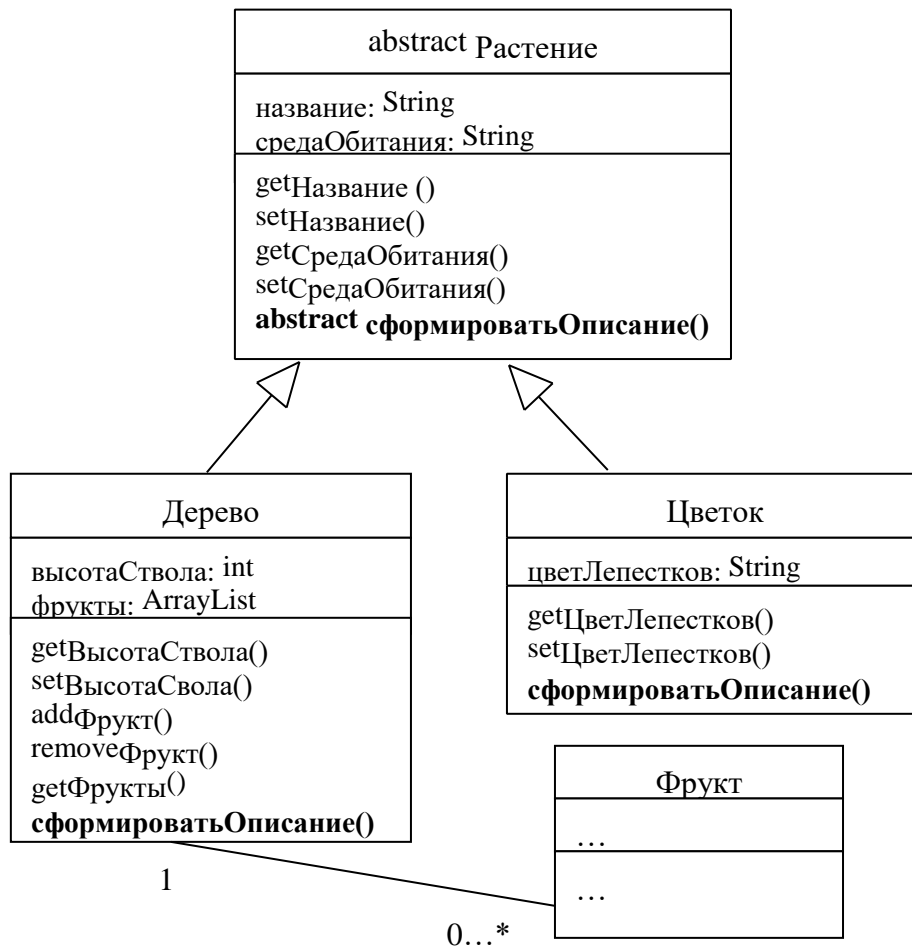
На дереве растут фрукты: <фрукт1>, ... <фруктN>»

Для цветка:

«Название цветка: <название>, Среда обитания: <среда обитания> Цвет

лепестков: <цвет лепестков>»

Если бы описание формировалось одинаково для Деревьев и Цветов (т.е. для всех Растений), то данная задача решалась бы путем реализации метода `String формироватьОписание()` в классе Растение. Однако в нашем случае ситуация несколько иная: *метод `String формироватьОписание()` относится ко всем Растениям, однако реализуется, причем по-разному в классах `Дерево` и `Цветок`.*



Если метод, относящийся к базовому классу, не может быть реализован в нем, но может быть реализован в дочерних классах, то данный метод объявляется **абстрактным**. Абстрактный метод содержит только заголовок, но не имеет содержимого (реализации).

Пример:

```

abstract class Plant {

    public abstract String sformirovatOpisanie();

    ...

}
  
```

При наличии абстрактного метода в базовом классе, все дочерние классы **потребуют** реализации данного метода. Т.е. дочерние (не абстрактные) классы должны «знать» что выполняет данный метод. Реализация метода, объявленного в базовом классе, называется **переопределение** метода (overriding). Иными словами, **переопределение** – это изменение в дочернем классе поведения метода, объявленного в базовом классе. При этом переопределяемый метод базового класса не обязательно должен быть абстрактным.

Для переопределения метода в дочернем классе следует объявить метод с такой же сигнатурой (т.е. названием, типом возвращаемого значения и набором параметров) как в базовом классе.

В нашем примере метод сформироватьОписание(), объявленный абстрактным в классе Растение, перегружается в классах Дерево и Цветок:

```

abstract class Plant {
    public abstract String sformirovatOpisanie(); ...
}
class Tree extends Plant {

    public String sformirovatOpisanie() {
  
```

```

        String s; s = "Название дерева:" +
        nazvanie + ", Среда обитания: " +
        sredaObitaniya +
        "Высота ствола:" + vysotaStvola + " м.," +
        "На дереве растут фрукты:";

        for (int i=0; i<fruits.size(); i++) {      s = s +
        ((Fruit)fruits.get(i)).getNazvanie() + "\n"; } return s;
    }
    ...
}

class Flower extends Plant {
    public String sformirovatOpisanie() {
        String s;
        s = "Название цветка:" + nazvanie + ", Среда
        обитания:" + sredaObitaniya +
        " Цвет лепестков:" + cvetLepestkov;
        return
        s;
    }
    ..
    .
}

```

4. Полиморфизм

Предположим, в программе имеется коллекция (ArrayList), содержащая объекты классов Дерево и Цветок и требуется вывести описания (вызвать метод сформироватьОписание) для всех объектов данной коллекции. Проблема состоит в том, что коллекция в данном случае содержит **разнотипные** объекты, приведенные к базовому типу Object, в результате чего при считывании очередного элемента коллекции возникает задача определения типа объекта. Определив точный тип объекта (в нашем примере Дерево или Цветок) мы можем обратиться к нему с помощью ссылки соответствующего типа и вызвать метод сформироватьОписание.

В принципе в JAVA имеется механизм определения типа объекта (RTTI), однако данный механизм достаточно сложен и имеется далеко не во всех объектно-ориентированных языках.

Другой, более простой и универсальный механизм работы с разнотипными объектами состоит в том, что **к объектам определенного типа можно обращаться через ссылку базового типа**. В нашем примере это означает, что к любому объекту-Дереву или объекту-Цветку можно обратиться с помощью ссылки типа Растение. При этом у объекта будут доступны только те методы, которые определены в базовом классе (т.е. методы класса Растение). Однако, **при обращении к объекту дочернего класса по ссылке базового, вызывая определенный метод, определенный в базовом классе и переопределенный в дочернем, произойдет вызов метода, реализованного в дочернем классе**. Здесь действует механизм полиморфизма, который «определяет» действительный тип объекта и вызывает метод, реализованный в классе, соответствующем типу данного объекта.

Пример:

Допустим, имеется коллекция p, содержащая объекты типа Дерево и типа Цветок.
Требуется вывести описания всех объектов, содержащихся в коллекции

```

for (int i=0; i<p.size(); i++) {

    // считываем очередной объект из коллекции и
    // обращаемся к нему по ссылке базового типа
    Plant plant = (Plant)p.get(i);

    // вызываем метод сформироватьОписание у объекта.
    // Несмотря на то, что обращение к объекту идет через ссылку типа
    // Растение, метод вызывается у Цветка или дерева, в зависимости
    // от того, какому типу принадлежит данный объект
    System.out.println(plant.sformirovatOpisanie());
}

```

ВАРИАНТЫ ЗАДАНИЙ

Для всех вариантов задания реализуются для

1. абстрактных классов

2. интерфейсов

Вариант № 1

Смоделировать структуру предприятия:

Классы	Свойства
Фирма	название (get, set)
Отдел	название (get, set) количество сотрудников (get, set)
Сотрудник	фио (get, set) должность (get, set) оклад (get, set)

Предположим, что все Сотрудники делятся на два типа:

- Штатный сотрудник
- Сотрудник по контракту

Требуется производить расчет заработной платы:

- Для Штатного сотрудника может быть установлена премия, тогда зар.плата = оклад + премия
- Для Сотрудника по контракту зар.плата = оклад

для этого выполнить следующие действия:

- сделать класс Сотрудник абстрактным (abstract)
- добавить в класс Сотрудник абстрактный метод «рассчитать зарплату»
- создать классы «Штатный сотрудник» и «Контрактник», унаследовать их от класса Сотрудник
- в класс Штатный сотрудник добавить свойство «премия» (set)
- перегрузить метод «рассчитать зарплату» в классах «Штатный сотрудник» и «Контрактник»

В один из отделов добавить двух сотрудников по контракту и двух штатных сотрудников. Назначить премии штатным сотрудникам. Используя цикл for рассчитать и вывести на экран зарплату по всем сотрудникам данного отдела.

Вариант № 2

Смоделировать структуру банка:

Классы	Свойства
Банк	название (get, set)
Филиал	название (get, set) общая сумма вкладов (get, set)
Вклад	фио вкладчика (get, set) сумма вклада (get, set)

Предположим, что все Вклады делятся на два типа:

- Долгосрочный - начисления процентов по вкладу зависят от суммы (до 10000 – 12%, больше 10000 – 18% годовых)
- До востребования (6% годовых независимо от суммы вклада)

Предполагаем, что проценты по вкладу начисляются каждый месяц. Требуется рассчитывать сумму вклада, по прошествии указанного количества месяцев.

для этого выполнить следующие действия: -

- сделать класс Вклад абстрактным (abstract)
- добавить в класс Вклад абстрактный метод «рассчитать сумму вклада» с параметром «количество месяцев»
- создать классы «Долгосрочный вклад» и «Вклад до востребования», унаследовать их от класса Вклад - перегрузить метод «рассчитать сумму вклада (количество месяцев)» в классах «Долгосрочный вклад» и «Вклад до востребования».

В определенный филиал добавить два долгосрочных вклада и три вклада до востребования. Используя цикл for рассчитать и вывести на экран первоначальную сумму и сумму по прошествии трех месяцев каждого вклада данного филиала.

Вариант № 3

Смоделировать структуру аэропорта:

Классы	Свойства
Аэропорт	название (get, set)
Летательный аппарат	название (get, set) макс. количество пассажиров (get, set)
Пассажир	фио (get, set) №посадочного места (get, set)

Предположим, что все Летательные аппараты делятся на два типа:

- Самолет
- Вертолет

Требуется реализовать процедуру взлета летательного аппарата, реализованную по разному для Самолета и Вертолета. При этом Самолет также отличается от Вертолета наличием свойства «длина полосы разгона».

Процедура выводит в текстовом виде примерно следующее:

- для Самолета: «<название л/а>. Разогреваю двигатели. Прохожу полосу разгона длиной <длина полосы разгона>. Взлетаю. Убираю шасси.»
- для Вертолета «<название л/а>. Разогреваю двигатели. Взлетаю.»

для этого выполнить следующие действия:

- сделать класс Летательный аппарат абстрактным (abstract)
- добавить в класс Летательный аппарат абстрактный метод «взлет»
- создать классы «Самолет» и «Вертолет», унаследовать их от класса Летательный аппарат - добавить в класс Самолет свойство «длина полосы разгона» (get, set)
- перегрузить метод «взлет» в классах «Самолет» и «Вертолет».

В определенный аэропорт добавить два самолета и один вертолет. Указать длину полосы разгона для самолетов. Используя цикл for вызвать метод «взлет» у каждого летательного аппарата аэропорта и вывести результаты на экран.

Вариант № 4

Смоделировать структуру библиотеки:

Классы	Свойства
Библиотека	название (get, set)
Отдел (по жанрам)	название жанра (get, set) количество изданий (get, set)
Издание	название (get, set) автор (get, set) год издания (get, set)

Предположим, что все Издания делятся на два типа:

- Книга – отличается наличием резюме (краткого описания)
- Журнал – отличается наличием множества статей

Требуется формировать текстовое описание Издания в виде:

- для Книги: «Автор: <автор>, год издания: <год издания> название: <название>
резюме: <резюме>»
- для Журнала: «Автор: <автор>, год издания: <год издания> название: <название>
статьи: <название статьи 1>, <название статьи 2> ... »

для этого выполнить следующие действия:

- сделать класс Издание абстрактным (abstract)
- добавить в класс Издание абстрактный метод «сформировать описание»
- создать классы «Книга» и «Журнал», унаследовать их от класса Издание - добавить в класс
Журнал множественное свойство «названия статей» (get, add)
- перегрузить метод «сформировать описание» в классах «Книга» и «Журнал».

В определенный отдел библиотеки добавить две книги и один журнал. В журнал добавить две статьи. Используя цикл for, сформировать и вывести на экран описание каждого издания данного отдела.

Вариант № 5

Смоделировать структуру компании сотовой связи:

Классы	Свойства
Компания	Название (get, set)
Тариф	название (get, set) количество абонентов (get, set)

Абонент	фио (get, set) номер телефона (get, set) остаток на счете (get, set)
---------	---

Предположим, что все Тарифы делятся на два типа:

- Посекундный. Отличается наличием свойства «стоимость секунды разговора» (get, set). Оплата производится за каждую секунду разговора.
- Поминутный. Отличается наличием свойства «стоимость минуты разговора» (get, set). Оплата производится за каждую минуту разговора, т.е., например, 20 сек. приравнивается 1 мин.

Требуется производить подсчет стоимости разговора указанной длительности (в секундах).

для этого выполнить следующие действия: -

- сделать класс Тариф абстрактным (abstract)
- добавить в класс Тариф абстрактный метод «рассчитать стоимость разговора» с параметром «количество секунд»
- создать классы «Посекундный тариф» и «Поминутный тариф», унаследовать их от класса Тариф
- в класс Посекундный тариф добавить свойство «стоимость секунды разговора» (get, set)
- в класс Поминутный тариф добавить свойство «стоимость минуты разговора» (get, set)
- перегрузить метод «рассчитать стоимость разговора (количество секунд)» в классах «Посекундный тариф» и «Поминутный тариф».

В определенную компанию добавить один поминутный тариф и один посекундный тариф. Указать стоимость минуты и секунды разговора соответственно. Используя цикл for, рассчитать и вывести на экран стоимость разговора продолжительностью 80 секунд для каждого тарифа данной компании.

Вариант № 6

Смоделировать структуру автосалона:

Классы	Свойства
Автосалон	название (get, set)
Автомобиль (марка)	название марки (get, set) макс. количество пассажиров (get, set) стоимость (get, set) количество на складе (get, set) boolean наличие (get, set)

Заявка на покупку	фио покупателя (get, set) номер телефона (get, set)
-------------------	---

Предположим, что все Заявки делятся на два типа:

- Заявки на приобретение со стенда.
- Заявки на отложенную поставку. Отличается наличием свойства «процент скидки» (get, set), действующей при приобретении автомобиля с отложенной поставкой.

Требуется производить подсчет стоимости заказа:

- Заявка на приобретение со стенда. Стоимость заказа = стоимость автомобиля, указанного в заявке.
- Заявка на отложенную поставку. Стоимость заказа = стоимость автомобиля, указанного в заявке * (100% - процент скидки).

для этого выполнить следующие действия: -

- сделать класс Заявка абстрактным (abstract)
- добавить в класс Заявка абстрактный метод «рассчитать стоимость заказа»
- создать классы «Заявка на приобретение со стенда» и «Заявка на отложенную поставку», унаследовать их от класса Заявка
- в класс Заявка на отложенную поставку добавить свойство «процент скидки» (get, set)
- перегрузить метод «рассчитать стоимость заказа» в классах «Заявка на приобретение со стенда» и «Заявка на отложенную поставку».

На определенный автомобиль оформить две заявки на приобретение со стенда, и одну заявку на отложенную поставку. Для заявки на отложенную поставку указать значение процента скидки. Используя цикл for, рассчитать и вывести на экран стоимость заказа по всем заявкам на данный автомобиль.

Вариант № 7

Смоделировать структуру музыкальной коллекции:

Классы	Свойства
Коллекция	название (get, set) фио владельца (get, set)

Музыкальный носитель (альбом)	автор/группа (get, set) жанр (get, set) год выпуска (get, set) общая продолжительность звучания (get, set)
Музыкальное произведение	название (get, set) продолжительность (get, set)

Предположим, что все Музыкальные произведения делятся на два типа:

- Песни – отличаются наличием свойств «текст» и «автор текста»
- Инструментальные произведения – отличаются наличием множественного свойства строкового типа «инструменты-участники»

Требуется формировать текстовое описание Музыкального произведения в виде:

- для Песни:
«Название: <название>,
продолжительность: <продолжительность>
текст: <текст>
автор текста: <автор текста>»
- для Инструментального произведения: «Название: <название>,
продолжительность: <продолжительность> инструменты-
участники: <инструменты-участники>»

для этого выполнить следующие действия:

- сделать класс Музыкальное произведение абстрактным (abstract)
- добавить в класс Музыкальное произведение абстрактный метод «сформировать описание»
- создать классы «Песня» и «Инструментальное произведение», унаследовать их от класса Музыкальное произведение
- добавить в класс Песня свойства «текст» и «автор текста» (set)
- добавить в класс Инструментальное произведение свойство «инструменты-участники» (add, remove) - перегрузить метод «сформировать описание» в классах «Песня» и «Инструментальное произведение».

В определенный альбом добавить две песни и одно инструментальное произведение. Для песен указать тексты песен и их авторов. Для инструментального произведения указать инструменты-участники. Используя цикл for, сформировать и вывести на экран описание каждого музыкального произведения данного альбома.

Смоделировать структуру реестра городского жилья:

Классы	Свойства
Город	название (get, set)
Здание	название улицы (get, set) номер дома (get, set) общая площадь (get) базовая ежемесячная оплата за кв.м площади (get, set)
Помещение	номер (get, set) площадь (get, set)

Предположим, что все Помещения делятся на два типа:

- Квартира – отличается наличием множественного свойства строкового типа «ФИО жильцов»
- Офис – отличается наличием свойств «Название фирмы-владельца», «Вид деятельности»

Требуется рассчитывать ежемесячную оплату за помещение по следующим правилам:

- для Квартиры:

Ежемес. оплата = базовая ежемесячная оплата за кв.м площади * площадь * (1 + кол-во жильцов*0,1)

- Для Офиса:

Ежемес. оплата = базовая ежемесячная оплата за кв.м площади * площадь * 2

для этого выполнить следующие действия:

- сделать класс Помещение абстрактным (abstract)
- добавить в класс Помещение абстрактный метод «рассчитатьЕжемесячнуюОплату»
- создать классы «Квартира» и «Офис», унаследовать их от класса Помещение
- добавить в класс Квартира свойство «ФИО жильцов» (add)
- добавить в класс Офис свойства «название фирмы-владельца» и «вид деятельности» (get, set)
- перегрузить метод «рассчитатьЕжемесячнуюОплату» в классах «Квартира» и «Офис».

В определенное здание добавить две квартиры и один офис. Для квартир указать ФИО жильцов. Для офиса указать название фирмы-владельца и вид деятельности. Используя цикл for, рассчитать и вывести на экран размер ежемесячной оплаты для каждого помещения в данном здании.

Смоделировать структуру зоопарка:

Классы	Свойства
Зоопарк	название (get, set)
Вольер/клетка	номер (get, set) размер (get, set) макс. количество животных (get, set) текущее количество животных (get, set)
Животное	название (get, set) boolean хищник (get, set)

Предположим, что все Животные, живущие в зоопарке, делятся на три типа:

- Рыбы – отличаются наличием свойства boolean «глубоководная»
- Птицы – отличаются наличием свойства «скорость полета»
- Звери – отличается наличием свойства «среда обитания» (н-р, «лес», «степь», «пустыня» и т.п.)

Требуется формировать описание животных по следующему правилу:

- для Рыбы:
«Класс: <глубоководная/не глубоководная> рыба,
Название: <название>,
хищник: <да/нет>»
- для Птицы:
«Класс: птица,
Название: <название>,
скорость полета: <скорость полета>»
- для Зверя:
«Класс: зверь,
Название: <название>, Среда
обитания: <среда обитания>»

для этого выполнить следующие действия:

- сделать класс Животное абстрактным (abstract)
- добавить в класс Животное абстрактный метод «сформировать описание»
- создать классы «Рыба», «Птица» и «Зверь», унаследовать их от класса Животное
- добавить в класс Рыба свойство «глубоководная» (set)
- добавить в класс Птица свойство «скорость полета» (set)
- добавить в класс Зверь свойство «среда обитания» (set)
- перегрузить метод «сформировать описание» в классах «Рыба», «Птица» и «Зверь».

В определенный зоопарк добавить одну рыбу, одну птицу и одного зверя. Для рыбы указать значение свойства «глубоководная», для птицы – «скорость полета», для зверя – «среда обитания». Используя цикл for, сформировать и вывести на экран описание каждого животного в зоопарке.

Вариант № 10

Смоделировать структуру автоматизированного банкомата:

Классы	Свойства
Банк	название (get, set)
Счет	номер (get, set) PIN-код (get, set) остаток (get, set)
Банкомат	идентификационный номер (get, set) адрес (get, set)

Предположим, что все Счета делятся на два типа:

- обычный
- льготный

Требуется реализовать метод «снять со счета(сумма)» так, чтобы при снятии определенной суммы с Обычного счета дополнительно взималось 5% от указанной суммы, а при снятии с Льготного счета – взимался всего 1%.

для этого выполнить следующие действия: -

сделать класс Счет абстрактным (abstract)

- в классе Счет сделать абстрактным метод «снять со счета» с параметром «сумма»
- создать классы «Обычный счет» и «Льготный счет», унаследовать их от класса Счет - перегрузить метод «снять со счета(сумма)» в классах «Обычный счет» и «Льготный счет».

В определенный банк добавить два обычных счета и два льготных счета и задать значения свойств.

Используя цикл for, вывести на экран остаток по каждому счету данного банка, затем снять с каждого счета определенную сумму и снова вывести остаток по каждому счету.