2. Объекты и классы: создание классов и методов классов.

Теория

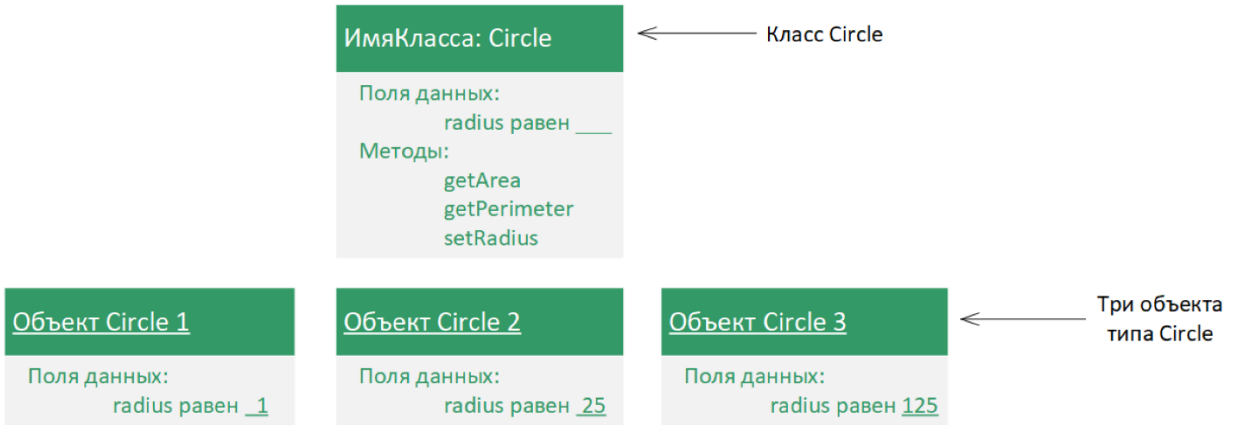
Объектно-ориентированное программирование по существу является технологией разработки многократно используемого программного обеспечения. Изучив материалы курсов «Основы Java-программирования», вы уже можете решить многие задачи программирования с помощью структур выбора, циклов, методов и массивов. Однако этих возможностей Java недостаточно для разработки крупномасштабных программных систем.

Класс определяет свойства и поведение объектов.

Объектно-ориентированное программирование (ООП) включает в себя программирование с помощью объектов. Объект представляет сущность реального мира, которая может быть четко идентифицирована. Например, слушатель, стол, круг, кнопка и даже кредит могут рассматриваться как объекты. Объект имеет уникальный идентификатор, состояние и поведение.

- Состояние объекта (также известное как его свойства или атрибуты) представлено полями данных с их текущими значениями. У объекта круга, например, есть поле данных radius, которое является свойством, характеризующим круг. У объекта прямоугольника, например, есть поля данных **width** и **height**, которые являются свойствами, характеризующими прямоугольник.
- Поведение объекта (также известное как его действия) определяется методами. Чтобы вызвать метод объекта, необходимо попросить объект выполнить действие. Например, можно определить методы **getArea()** и **getPerimeter()** для объектов круга. У объекта круга можно вызвать метод **getArea()**, чтобы вернуть его площадь, и **getPerimeter()**, чтобы вернуть его периметр. Можно также определить метод **setRadius(radius)**. У объекта круга можно вызвать этот метод, чтобы изменить его радиус.

Объекты одного типа определяются с помощью общего класса. Класс — это шаблон, образец или проект, который определяет поля данных и методы объекта. Объект является экземпляром класса. Можно создать несколько экземпляров одного класса. Создание экземпляра называется инстанцированием. Термины «объект» и «экземпляр класса» часто являются взаимозаменяемыми. Отношения между классом и объектами аналогичны отношениям между рецептом яблочного пирога и самим яблочным пирогом: по одному рецепту вы можете приготовить сколько угодно яблочных пирогов. На следующем рисунке показан класс **Circle** и три его объекта.



Java-класс использует переменные для определения полей данных, а методы — для определения действий. Кроме того, класс предоставляет методы специального типа, известные как конструкторы, которые вызываются для создания нового объекта. Конструктор может выполнять любое действие, но предназначен он для выполнения инициализирующих действий, таких как инициализация полей данных объектов.

Конструктор — это особый вид метода. У него есть три особенности:

- Конструктор должен иметь то же имя, что и класс.
- Конструктор не имеет типа возвращаемого значения — даже типа void.
- Конструктор вызывается при создании объекта с помощью оператора new, таким образом он играет роль инициализатора объектов.

Конструктор имеет точно такое же имя, как и определяющий его класс. Как и обычные методы, конструкторы могут быть перегружены (т.е. несколько конструкторов могут иметь одно и то же имя, но разные сигнатуры), что упрощает создание объектов с разными начальными значениями полей данных.

```
public void Circle() { }
```

В этом случае **Circle()** является методом, а не конструктором.

Конструкторы используются для создания объектов. Чтобы создать объект класса, вызовите конструктор класса с помощью оператора new следующим образом: `new ИмяКласса(аргументы);`

Класс можно определить и без конструкторов. В таком случае в классе неявно определяется общедоступный конструктор без аргументов с пустым телом. Этот конструктор, называемый заданным по умолчанию конструктором, предоставляется автоматически, только если в классе нет явно определенных конструкторов.

Задача #1

Напишите программу, в которой создается класс Car. В данном классе должны быть обозначены следующие поля: String model, String license, String color, int year – модель автомобиля, номер автомобиля, цвет автомобиля и год выпуска соответственно. Класс должен содержать три конструктора, один конструктор, который включает в себя все поля класса, один конструктор по умолчанию, один включает поля по выбору студента.

Задача #2

В отдельном классе Main создайте экземпляры классов (объекты), используя различные конструкторы, реализованные в задаче #1. Создайте в классе метод To_String(), который будет выводить значения полей экземпляров класса. Проверьте работу созданного метода, вызвав его у объекта. Дополните класс методами для получения и установки значений для всех полей (геттерами и сеттерами). Создайте метод класса, который будет возвращать возраст автомобиля, вычисляющийся от текущего года, значение текущего года допускается сделаться константным.