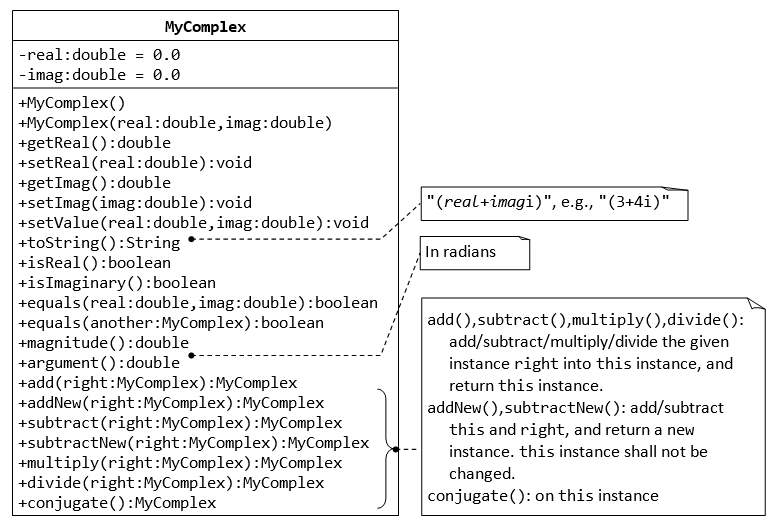
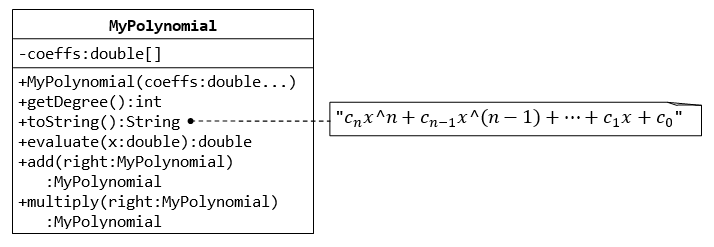
**Задачи на тему: основные приницпы объектно ориентированного программирования, часть 2**

**1. Реализовать класс для работы с комплексными числами**



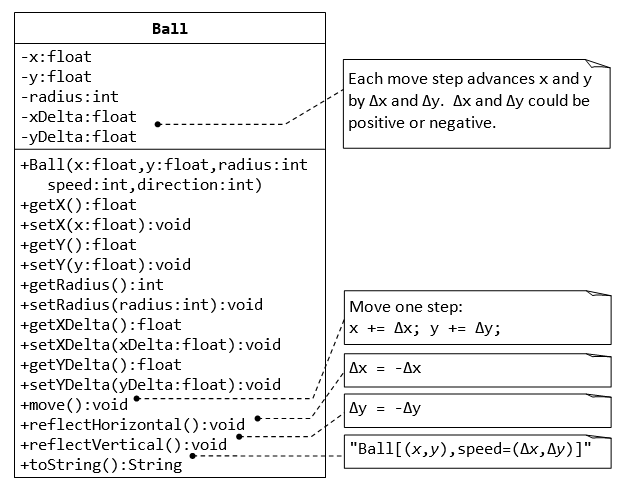
* Конструктор по умолчанию MyComplex создаёт число вида 0.0 + 0.0i
* isReal() и isImaginary() проверяют содержит ли число действительную и мнимую части.
* Необходимо переопределить метод equals таким образом, чтобы он возвращал true если действительная и мнимая части двух комплексных чисел совпадают.
* Метод magnitude() возвращает модуль комплексного числа
* Метод argument() возвращает аргумент комплексного числа, в радианах
* Методы addNew(MyComplex right) и subtractNew(MyComplex right) складывают/вычитают с/из текущего числа (this объект) и результат вычисления возвращается в качестве нового созданного объекта MyComplex (т.е. поля в this не меняются, а создаётся новый объект)
* Метод conjugate() возвращает объект MyComplex комплексного числа, сопряжённого с текущим (т.е. c this)

**2. Реализовать класс для работы с полиномами**

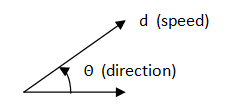


* Конструктор с переменым количеством аргументов, создаёт полином степени n. Для того чтобы создать полином степени n, конструктору необходимо передать на вход массив размера n+1, коэффициенту c0 соотвествует индекс 0
* Метод getDegree() возвращает стпенеь полинома
* Метод toString() возвращает строкове представление объекта полинома в виде cnx^n+cn-1x^(n-1)+...+c1x+c0
* Методы add() и multiple() складывают/умножают текущий полином (this) с заданным, и в качестве результата возвращают новый объект класса MyPolynomial

**3.Реализовать класс, моделирующий перемещение мяча**



* Переменные x,y соотвествуют координатам центра мяча, radius – радиус мяча
* xDelta (Δx) and yDelta (Δy) – соотвествуют перемещению мяча на заданный шаг (могут принаимать положительные и отриуательные занчения)
* В конструктор передаются значения x, y, radius, скорость перемещения speed и направление перемещения direction (принимает занчения от -1800  до +1800  ) См. рисунок ниже



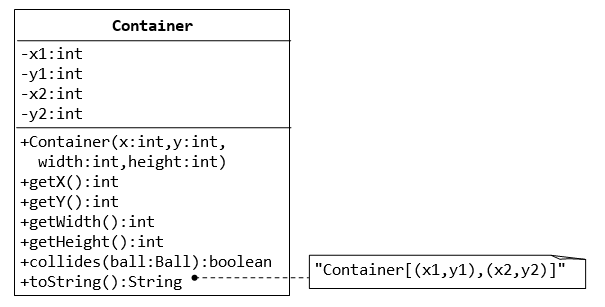
* Скорость и перемщение мяча связаны между следующими соотношениями:

Δx = d \* cos(ϑ)

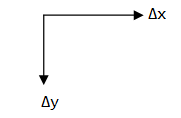
Δy = - d \* sin(ϑ)

* Метод move() перемещеат мячик на шаг x=x+Δx, y = y+Δy (или кратко можно так записать x += Δx)
* Метод reflectHorizontal() меняет направление перемещения мяча по горизонтали на противоположное (мячик отскакивает от горизонтальной стенки), следующим образом: Δx=- Δx, Δy не меняется
* Метод reflectVertical() меняет направление мяча по вертикали на противоположное (мячик отскакивает от вертикальной стенки), следующим образом: Δy=- Δy, Δx не меняется

Для тестирования написанного класса, создадим вспомогательный класс **Container**



* Класс Container представляет собой прямоугольный контейнер или ящик куда помещается мячик. Начало отсчёта с точкой (0,0) располоагется в верхнем левом углу:



* Координаты (x1,y1) и (x2,y2) соотвествуют верхнему левому и правому нижнему углу контейнрера.
* Конструктор принимает в качестве входных аргументов координаты (x,y) верхнего левого угла, ширину width и высоту height на основании этих занчений вычисляются координаты (x2,y2)
* Метод collidesWidth(Ball ball) проверяет находится ли данный мяч в границах контейнера или нет (с учётом radius) и возвращает true или false

В классе содержащем метод main необходимо протетисровать функциональность Ball и Container путём вызова методов move(), collidesWidth() и т.д.