Зертханалық жұмыс - №3

Объектіге бағытталған программалаудың элементтері

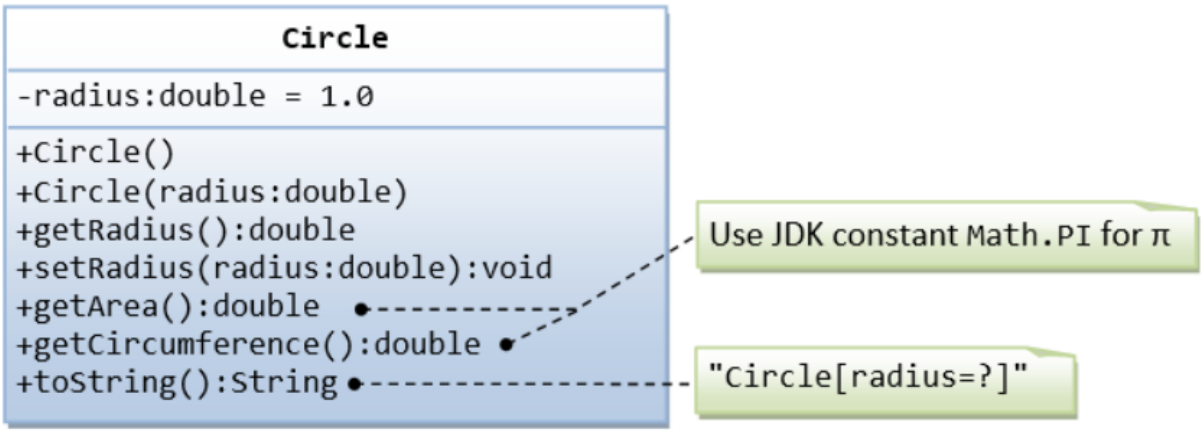
Жұмыстың мақсаты: Python программалау тілінде объектіге бағытталған программалау элементімен танысу

In [ ]:
```
from IPython import display
```

Класс құруға мысал

In [ ]:
```
display.Image('oop-1.png')
```

Out[ ]:



In [ ]:

In [ ]:
```
class Circle:
    def __init__(self, *args):
        if len(args) > 0:
            if isinstance(args[0], int) or isinstance(args[0], float):
                self.__radius = float(args[0])
        elif len(args) == 0:
            self.__radius = 1.0

    def getRadius(self)->float:
        return self.__radius

    def setRadius(self, radius:float):
        if radius > 0 and (isinstance(radius, float) or isinstance(radius, int)):
            self.__radius = float(radius)

    def getArea(self)->float:
        return 3.1415*self.__radius**2

    def getCircumference(self)->float:
        return 2*3.1415*self.__radius

    def __str__(self):
        return "Circle[radius = " + str(self.__radius) + "]"
```

In [ ]:
```
c1 = Circle(1.1)
print(c1)
```

```
Circle[radius = 1.1]
```

In [ ]:
```
c2 = Circle()
print(c2)
```

```
Circle[radius = 1.0]
```

In [ ]:
```
c1.setRadius(2.2)
print(c1)
print(f"radius is:  {c1.getRadius()}")
```

```
Circle[radius = 2.2]
radius is:  2.2
```

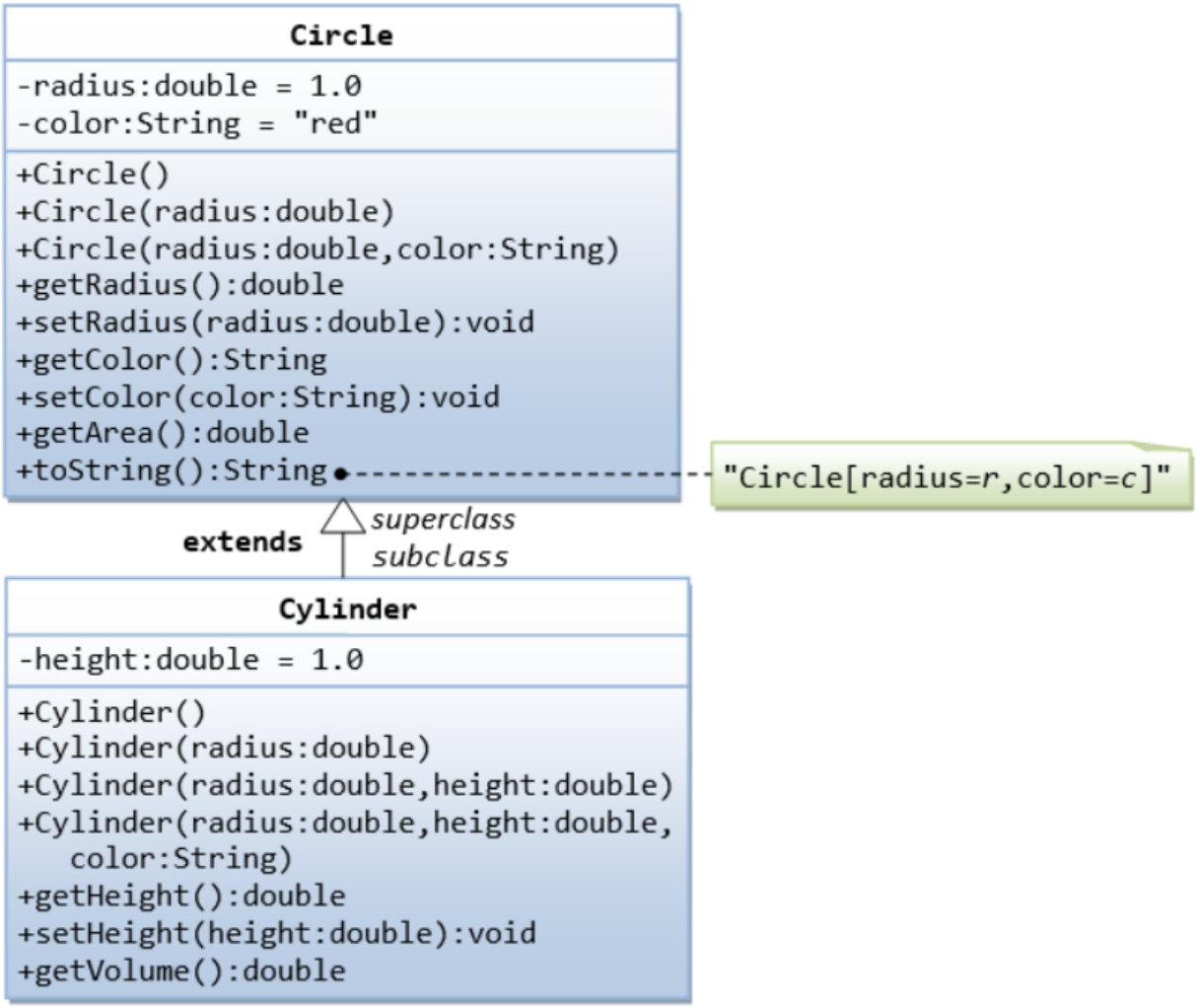
In [ ]:
```
print(f"area is: {c1.getArea():.2f}")
print(f'circumference is: {c1.getCircumference():.2f}')
```

```
area is: 15.20
circumference is: 13.82
```

ООП-ның мұрагерлік қасиетіне мысал

In [ ]:
```
display.Image('oop-8.png')
```

Out[ ]:



In [ ]:
```
class Circle2:
    def __init__(self, *args):
        if len(args) == 0:
            self.__radius = 1.0
            self.__color = "red"
        elif len(args) == 1:
            if isinstance(args[0], int) or isinstance(args[0], float):
                self.__radius = float(args[0])
                self.__color = "red"
        elif len(args) == 2:
            if (isinstance(args[0], int) or isinstance(args[0], float)) \
                and isinstance(args[1], str):
                    self.__radius = float(args[0])
                    self.__color = args[1]

    def getRadius(self)->float:
        return self.__radius

    def setRadius(self, radius:float):
        if radius > 0 and (isinstance(radius, float) or isinstance(radius, int)):
            self.__radius = float(radius)

    def getArea(self)->float:
        return 3.1415*self.__radius**2

    def getColor(self)->str:
        return self.__color

    def setColor(self, color:str):
        if isinstance(color, str):
            self.__color = color

    def __str__(self):
        return "Circle[radius = " + str(self.__radius) + ", color = " + self.__color
```

In [ ]:
```
class Cylinder(Circle2):
    def __init__(self, *args):
        if len(args) == 0:
            super().__init__()
            self.__height = 1.0
        elif len(args) == 1:
            super().__init__(args[0])
            self.__height = 1.0
        elif len(args) == 2 and isinstance(args[1], float):
            super().__init__(args[0])
            self.__height = args[1]
        elif len(args) == 3 and isinstance(args[1], float) \
            and isinstance(args[2], str):
            super().__init__(args[0], args[2])
            self.__height = args[1]
        else:
            raise ValueError
    def getHeight(self)->float:
        return self.__height

    def setHeight(self, height:float):
        if isinstance(height, float):
            self.__height = height

    def getVolume(self)->float:
        return super().getArea() * self.__height

    def __str__(self):
```

```
        return "Cylinder: sunclass of " + super().__str__() + \
            " height = " + str(self.__height)
```

In [ ]:

```
c1 = Cylinder()
print(f"radius = {c1.getRadius()}")
print(f"height = {c1.getHeight()}")
print(f"area = {c1.getArea()}")
print(f"volume = {c1.getVolume()}")
print(c1)
```

```
radius = 1.0
height = 1.0
area = 3.1415
volume = 3.1415
Cylinder: sunclass of Circle[radius = 1.0, color = red] height = 1.0
```

In [ ]:

```
c2 = Cylinder(10.0)
print(f"radius = {c2.getRadius()}")
print(f"height = {c2.getHeight()}")
print(f"area = {c2.getArea()}")
print(f"volume = {c2.getVolume()}")
print(c2)
```

```
radius = 10.0
height = 1.0
area = 314.15000000000003
volume = 314.15000000000003
Cylinder: sunclass of Circle[radius = 10.0, color = red] height = 1.0
```

In [ ]:

```
c3 = Cylinder(2.0, 10.0)
print(f"radius = {c3.getRadius()}")
print(f"height = {c3.getHeight()}")
print(f"area = {c3.getArea()}")
print(f"volume = {c3.getVolume()}")
print(c3)
```

```
radius = 2.0
height = 10.0
area = 12.566
volume = 125.66000000000001
Cylinder: sunclass of Circle[radius = 2.0, color = red] height = 10.0
```

In [ ]:

```
c4 = Cylinder(2.0, 10.0, "blue")
print(f"radius = {c4.getRadius()}")
print(f"height = {c4.getHeight()}")
print(f"area = {c4.getArea()}")
print(f"volume = {c4.getVolume()}")
print(f"color = {c4.getColor()}")
print(c4)
```

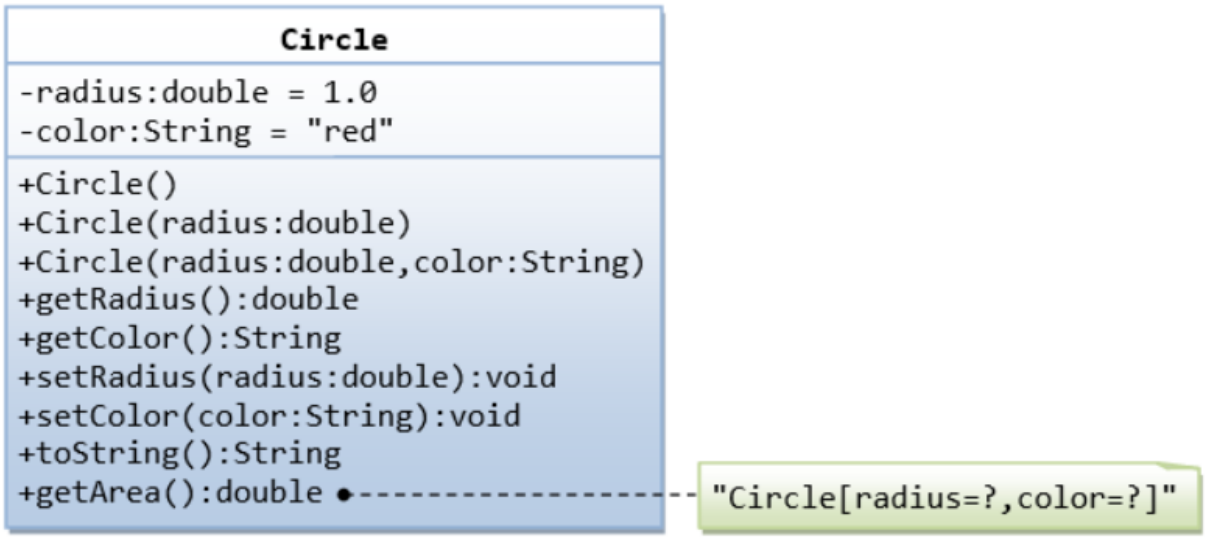
```
radius = 2.0
height = 10.0
area = 12.566
volume = 125.66000000000001
color = blue
Cylinder: sunclass of Circle[radius = 2.0, color = blue] height = 10.0
```

Тапсырмалар:

Төмендегі класс диаграммаларға қарап python тілінде класстар құрыңыздар

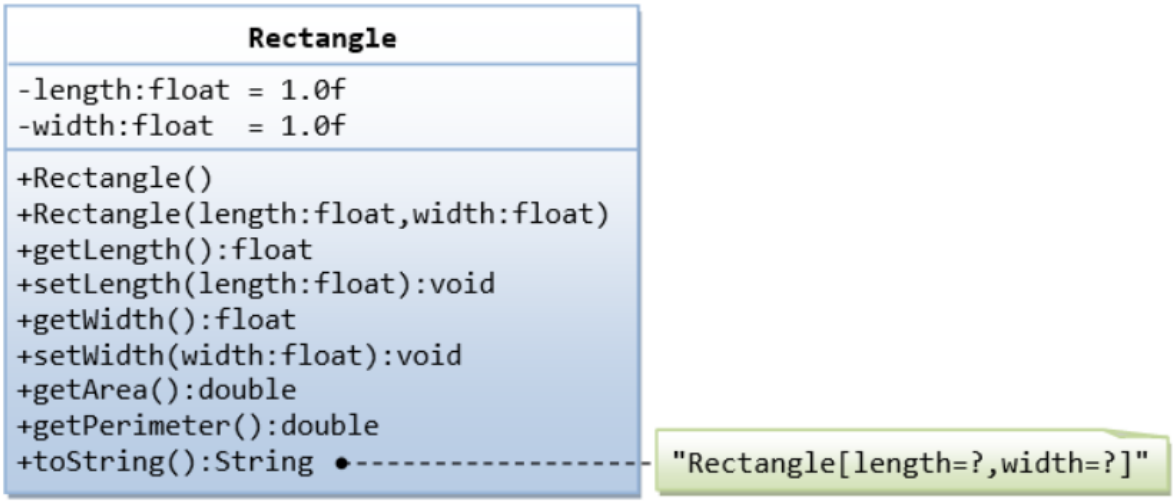
In [ ]:
```
display.Image('oop-2.png')
```

Out[ ]:



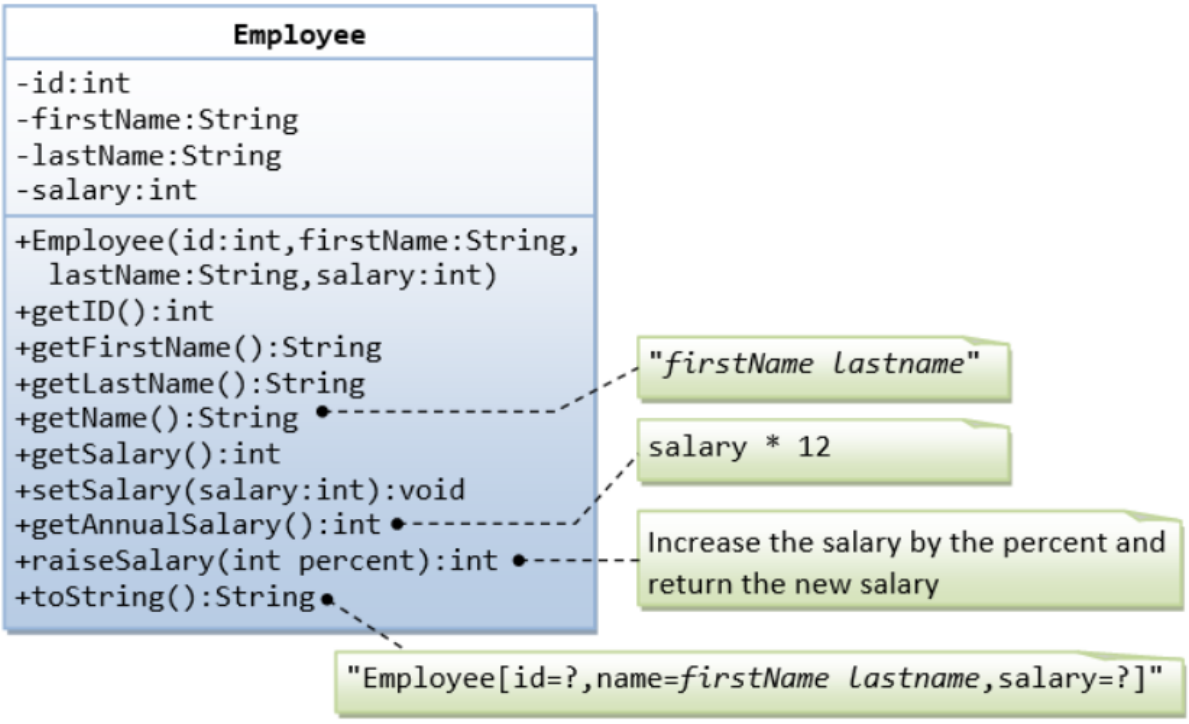
In [ ]:
```
display.Image('oop-3.png')
```

Out[ ]:



In [ ]:
```
display.Image('oop-4.png')
```

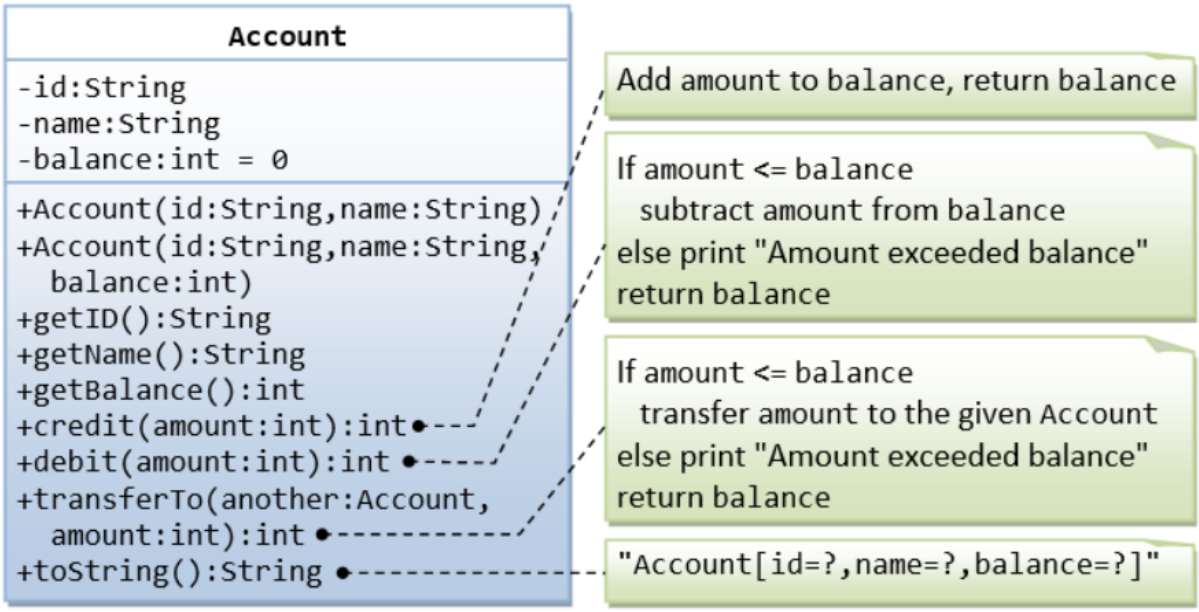
Out[ ]:



In [ ]:

```
display.Image('oop-5.png')
```

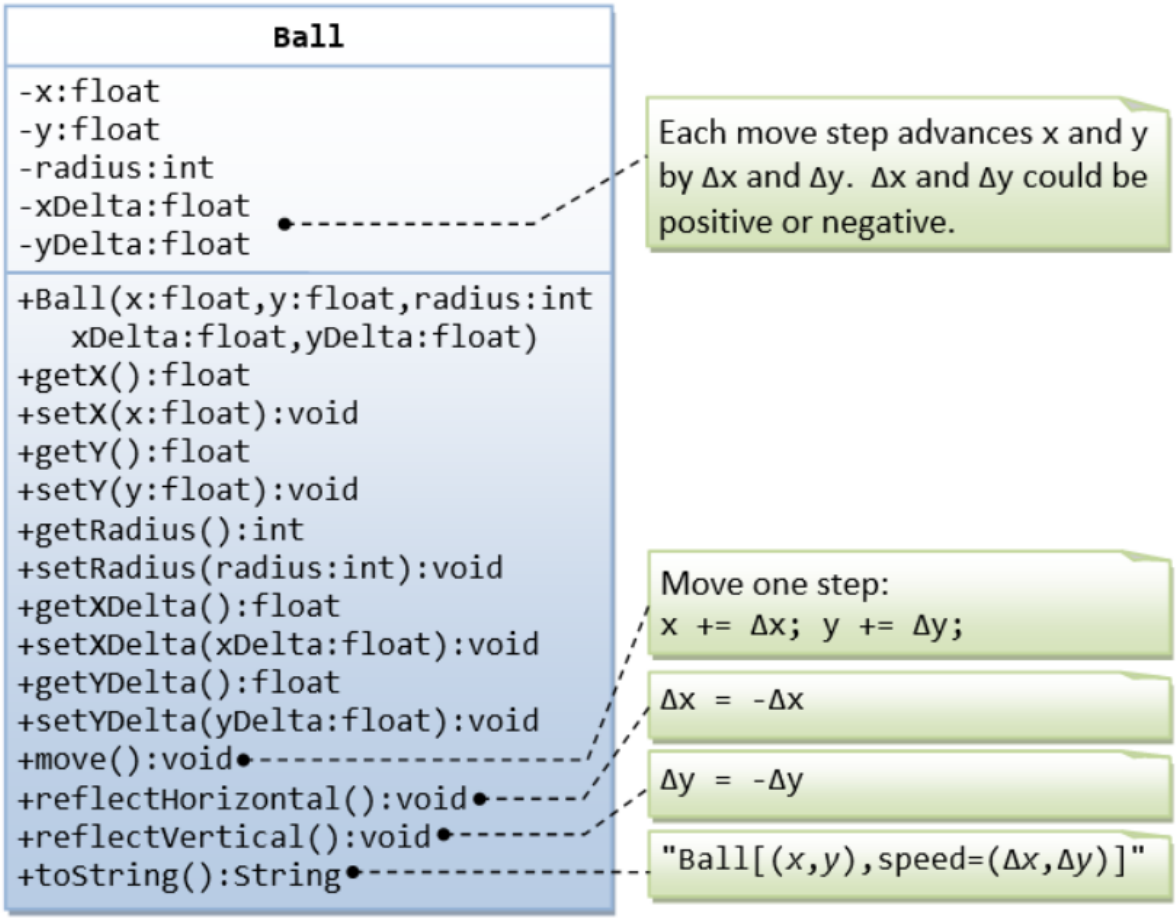
Out[ ]:



In [ ]:

```
display.Image('oop-6.png')
```

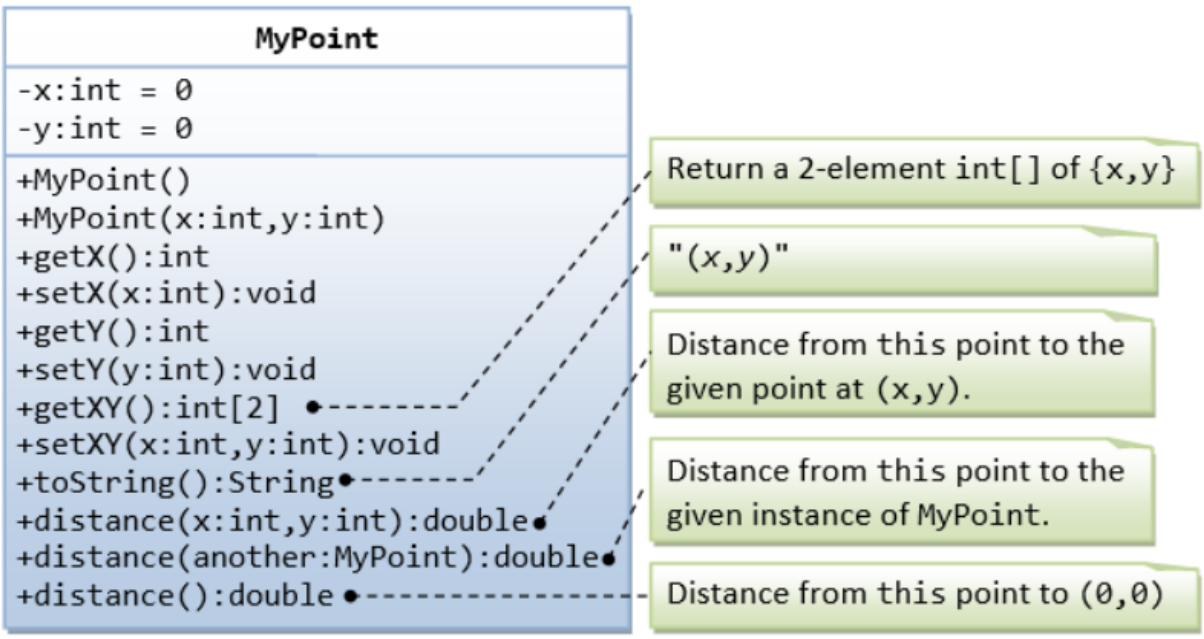
Out[ ]:



In [ ]:

```
display.Image('oop-7.png')
```

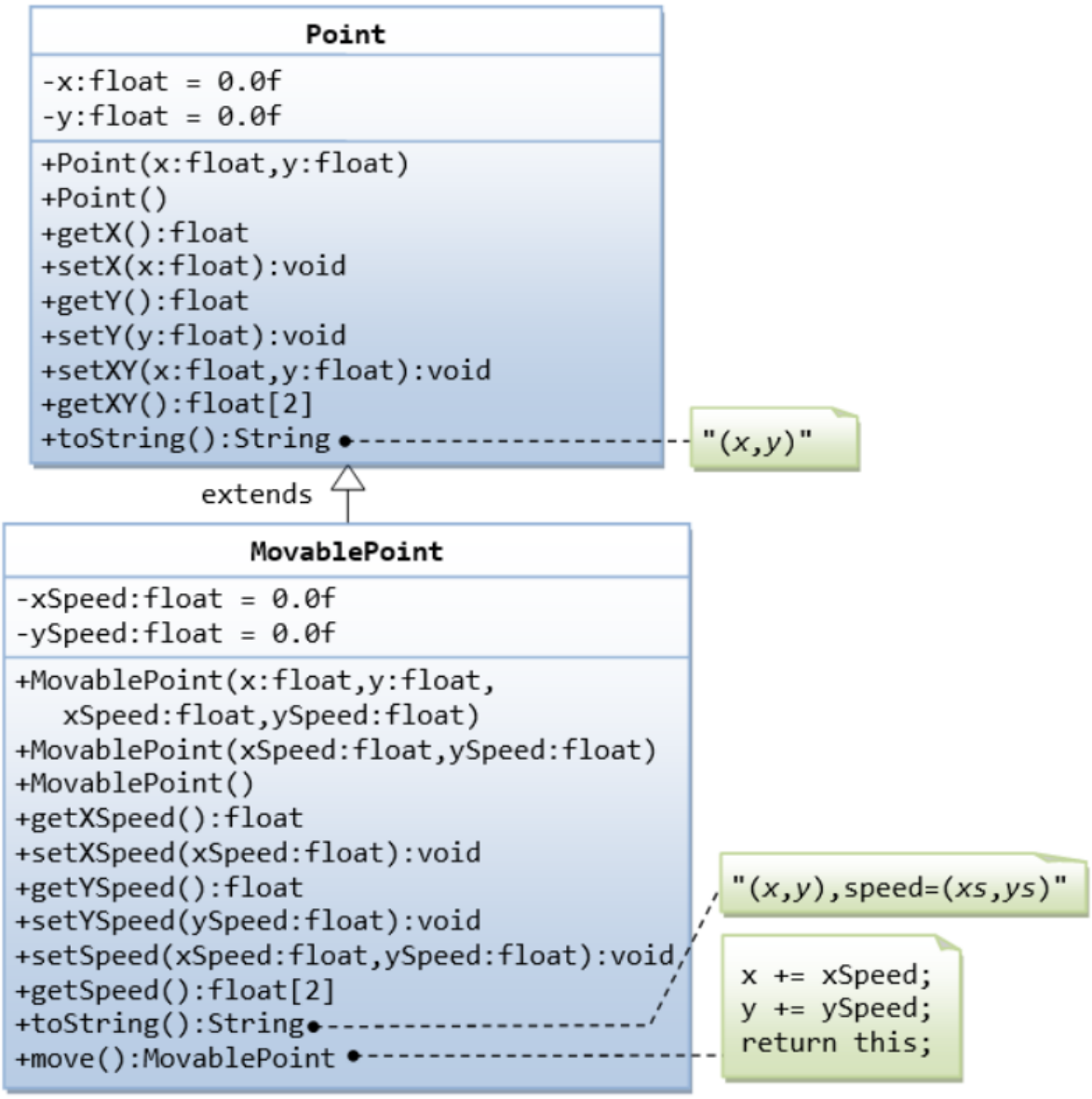
Out[ ]:



In [ ]:

```
display.Image('oop-9.png')
```

Out[ ]:


Point
-x:float = 0.0f
-y:float = 0.0f
+Point(x:float,y:float)
+Point()
+getX():float
+setX(x:float):void
+getY():float
+setY(y:float):void
+setXY(x:float,y:float):void
+getXY():float[2]
+toString():String
"(x,y)"
extends
MovablePoint
-xSpeed:float = 0.0f
-ySpeed:float = 0.0f
+MovablePoint(x:float,y:float,
xSpeed:float,ySpeed:float)
+MovablePoint(xSpeed:float,ySpeed:float)
+MovablePoint()
+getXSpeed():float
+setXSpeed(xSpeed:float):void
+getYSpeed():float
+setYSpeed(ySpeed:float):void
+setSpeed(xSpeed:float,ySpeed:float):void
+getSpeed():float[2]
+toString():String
+move():MovablePoint
"(x,y),speed=(xs,ys)"
x += xSpeed;
y += ySpeed;
return this;

In [ ]: