Napisz klasę MyTime, według diagramu umieszczonego poniżej.

Klasa powinna zawierać trzy pola prywatne: hours (0-24), minutes (0-59), seconds (0-59)

```
MyTime
-hour:int = 0
-minute:int = 0
-second:int = 0
+MyTime()
+MyTime(hour:int,minute:int,second:int)
+setTime(hour:int,minute:int,second:int):void
+getHour():int
+getMinute():int
+getSecond():int
+setHour(hour:int):void
+setMinute(minute:int):void
+setSecond(second:int):void
                                                   "HH:MM:SS"
+toString():String ◆--
                                                   with leading zeros,
+nextSecond():MyTime
                                                   e.g., "14:01:09"
+nextMinute():MyTime
+nextHour():MyTime
+previousSecond():MyTime
+previousMinute():MyTime
+previousHour():MyTime
```

### Oraz:

- 2 konstruktory (bez i sparametryzowany),
- Gettery i settery dla pól prywatnych
- Nadpisaną metodę toString(),
- Metody:
- nextSecond(), nextMinute(), nextHour() zwracające kolejno obiekt czasowy powiększony o
  czas wynikający z nazwy metody,
- prevSecond(), prevMinute(), prevHour() zwracające kolejno obiekt czasowy pomniejszony o
  czas wynikający z nazwy metody

Napisz klasę MyDate, według schematu poniżej:

```
MyDate
-year:int
-month:int
-day:int
-strMonths:String[] =
    "Jan","Feb","Mar","Apr","May","Jun",
"Jul","Aug","Sep","Oct","Nov","Dec"}
-strDays:String[] =
   {"Sunday", "Monday", "Tuesday", "Wednesday",
    "Thursday", "Friday", "Saturday"}
-daysInMonths:int[] =
   {31,28,31,30,31,30,31,30,31,30,31}
+isLeapYear(year:int):boolean
+isValidDate(year:int,month:int,day:int):boolean
+getDayOfWeek(year:int,month:int,day:int):int
+MyDate(year:int,month:int,day:int)
+setDate(year:int,month:int, day:int):void
+getYear():int
+getMonth():int
+getDay():int
+setYear(year:int):void
+setMonth(month:int):void
+setDay(day:int):void
+toString():String ◆-----
                                                          "xxxday d mmm yyyy"
+nextDay():MyDate
                                                         e.g., "Tuesday 14 Feb 2012"
+nextMonth():MyDate
+nextYear():MyDate
+previousDay():MyDate
+previousMonth():MyDate
+previousYear():MyDate
```

Wszystkie metody powinny sprawdzać poprawność przekazywanych danych, tak aby niemożliwym było ustalenie czasu, który nie istnieje (przykładowo 29 lutego dla roku nieprzestępnego, czy 25 godzina lub 66 sekunda).

Do napisania metody getDayOfWeek() użyj:

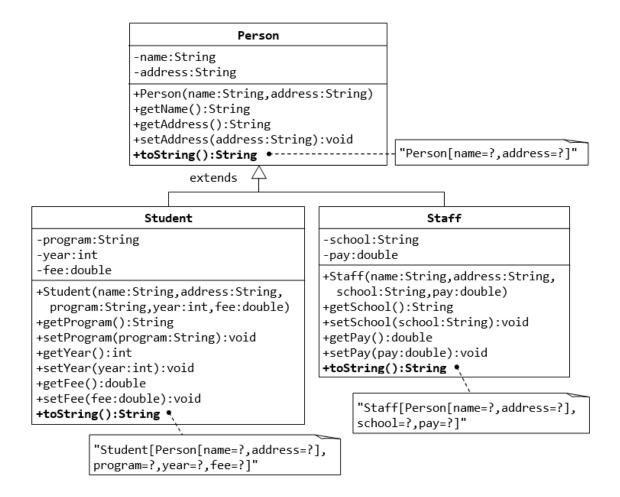
 $https://en.wikipedia.org/wiki/Determination\_of\_the\_day\_of\_the\_week\#A\_tabular\_method\_to\_calculate\_the\_day\_of\_the\_week\\$ 

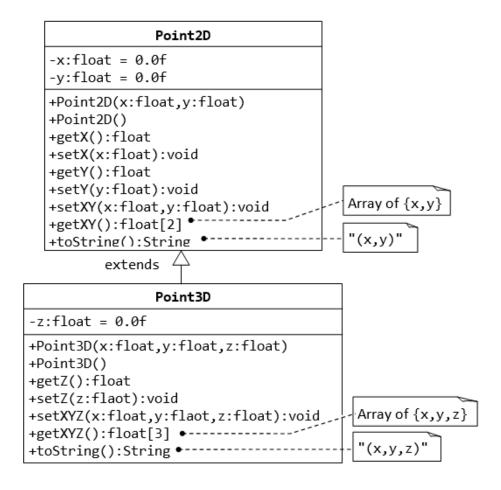
```
Circle
-radius:double = 1.0
-color:String = "red"
+Circle()
+Circle(radius:double)
+Circle(radius:double,color:String)
+getRadius():double
+setRadius(radius:double):void
+getColor():String
+setColor(color:String):void
+getArea():double
+toString():String •
                                           "Circle[radius=r,color=c]"
                    \ superclass
          extends
                     subclass
                Cylinder
-height:double = 1.0
+Cylinder()
+Cylinder(radius:double)
+Cylinder(radius:double,height:double)
+Cylinder(radius:double,height:double,
   color:String)
+getHeight():double
+setHeight(height:double):void
+getVolume():double
```

Napisz klasy Circle oraz Cylinder. Klasa Cylinder powinna być rozszerzeniem klasy Circle.

W poniższych ćwiczeniach, stwórz klasy tak jak pokazano na diagramach:

# **Ćwiczenie 4**





```
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]
                                             "(x,y)
 +toString():String
            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:flaot):void
                                               "(x,y),speed=(xs,ys)"
+getYSpeed():float
+setYSpeed(ySpeed:flaot):void
+setSpeed(xSpeed:float,ySpeed:flaot):void
                                                x += xSpeed;
+getSpeed():float[2]
                                                y += ySpeed;
+toString():String •
                                                return this;
+move():MovablePoint
```

# Shape -color:String = "red" -filled:boolean = true +Shape() +Shape(color:String, filled:boolean) +getColor():String +setColor(color:String):void +isFilled():boolean +setFilled(filled:boolean):void +toString():String

### Circle

- -radius:double = 1.0
- +Circle()
- +Circle(radius:double)
- +Circle(radius:double,
  - color:String,filled:boolean)
- +getRadius():double
- +setRadius(radius:double):void
- +getArea():double
  +getPerimeter():double
- +toString():String

### Rectangle

- -width:double = 1.0
- -length:double = 1.0
- +Rectangle()
- +Rectangle(width:double,
  - length:double)
- +Rectangle(width:double,
  - length:double,
  - color:String,filled:boolean)
- +getWidth():double
- +setWidth(width:double):void
- +getLength():double
- +setLength(legnth:double):void
- +getArea():double
- +getPerimeter():double
- +toString():String

# 7

### Square

- +Square()
- +Square(side:double)
- +Square(side:double,
  - color:String,filled:boolean)
- +getSide():double
- +setSide(side:double):void
- +setWidth(side:double):void
- +setLength(side:double):void
- +toString():String