Exercise Two: Classes and Objects

1. (The Fan class) (30 Points) Design a class named Fan to represent a fan. The class contains:

- Three constants named SLOW, MEDIUM, and FAST with the values 1, 2 and 3 to denote the fan speed.
- A private int data field named speed that specifies the speed of the fan (the default is SLOW).
- A private boolean data field named on that specifies whether the fan is on (the default is false).
- A private double date field named radius that specifies the radius of the fan (the default is 5).
- A string data field named color that specifies the color of the fan (the default is blue).
- The accessor and mutator methods for all four data fields.
- A no-arg constructor that creates a default fan.
- A method named toString() that returns a string description for the fan. If the fan is on, the method returns the fan speed, color, and radius in one combined string. If the fan is not on, the method returns the fan color and radius along with the string "Fan is off" in one combined string.

Implement the class and write a test program that creates two Fan objects. Assign maximum speed, radius 10 color yellow, and turn it on to the first object. Assign medium speed, radius 5, color blue, and turn it off to the second object. Display the objects by invoking their toString() method.

2. (The MyInteger class) (30 Points) Design a class named MyInteger. The class contains:

- An int data field named value that stores the int value represented by this object.
- A constructor that creates a MyInteger object for the specified int value.
- A getter method that returns the int value.
- The methods isEven() , isOdd(), and isPrime() that return true if the value in this object is even, odd, or prime, respectively.
- The static methods isEven(int), isOdd(int), and isPrime(int) that return true if the specified value is even, odd, or prime, respectively.
- The static methods isEven(MyInteger), isOdd(MyInteger), and isPrime(MyInteger) that return true if the specified value is even, odd, or prime, respectively.
- The methods equals(int) and equals(MyInteger) that return true if the value in this object is equals to the specified value.

Implement the class and write a test program that tests all methods in the class.

3. (Geometry: the Circle2D class) (40 Points) Define the Circle2D class that contains:

- Two double data fields named x and y that specify the center of the circle with getter methods.
- A data field radius with a getter method.
- A no-arg constructor that creates a default circle with (0,0) for (x, y) and 1 for radius.
- A constructor that creates a circle with the specified x, y, and radius.
- A method getArea() that returns the area of the circle.
- A method getPerimeter() that returns the perimeter of the circle.
- A method contains(double x, double y) that returns true if the specified point(x, y) is inside this circle (see Figure 1.a).
- A method contains(Circle2D circle) that returns true if the specified circle is inside this circle (see Figure 1.b).
- A method overlaps(Circle2D circle) that returns true if the specified circle overlaps with this circle(see Figure 1.c)

Implement the class, and write a test program that creates a Circle2D object c1 (new Circle2D(2, 2, 5.5)), displays its area and perimeter, and displays the result of c1.contains(3, 3), c1.contains(new Circle2D(4, 5, 10.5)), and c1.overlaps(new Circle2D(3, 5, 2.3)).

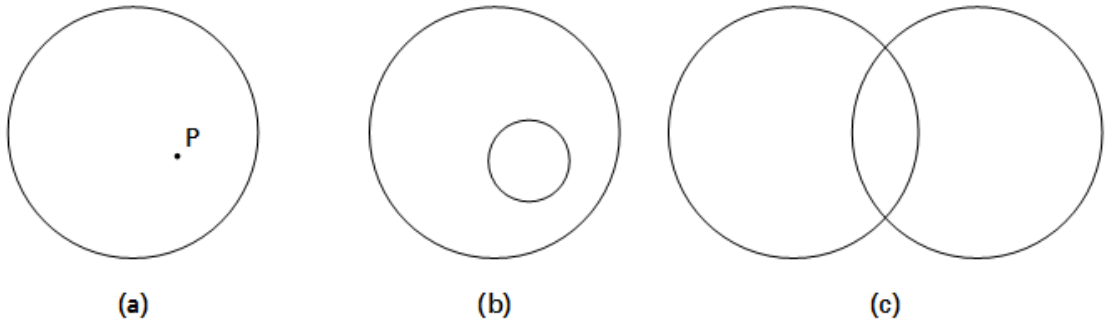

Figure 1：(a) A point in inside the circle. (b) A circle is inside another circle. (c) A circle overlaps another circle.

4. (The MyDate Classe) (Bonus Question: 20 Points) A class called MyDate, which models a date instance, is defined as shown in the class diagram (Figure 2).

The MyDate class contains the following private instance variables:

- year (int): Between 1 to 9999.
- month (int): Between 1 (Jan) to 12 (Dec).
- day (int): Between 1 to 28|29|30|31, where the last day depends on the month and whether it is a leap year for Feb (28|29).

It also contains the following private static variables (drawn with underlined in the class diagram):

- strMonths (String[]), strDays (String[]), and dayInMonths (int[]): static variables, initialized as shown, which are used in the methods.

The MyDate class has the following public static methods (drawn with underlined in the class diagram):

- isLeapYear(int year): returns true if the given year is a leap year. A year is a leap year if it is divisible by 4 but not by 100, or it is divisible by 400.
- isValidDate(int year, int month, int day): returns true if the given year, month, and day constitute a valid date. Assume that year is between 1 and 9999, month is between 1 (Jan) to 12 (Dec) and day shall be between 1 and 28|29|30|31 depending on the month and whether it is a leap year on Feb.
- getDayOfWeek(int year, int month, int day): returns the day of the week, where 0 for Sun, 1 for Mon, ..., 6 for Sat, for the given date. Assume that the date is valid. Read the Wiki "Determination of the day of the week" to learn how to determine the day of the week.

The MyDate class has one constructor, which takes 3 parameters: year, month and day. It shall invoke setDate() method (to be described later) to set the instance variables.

The MyDate class has the following public methods:

- setDate(int year, int month, int day): It shall invoke the static method isValidDate() to verify that the given year, month and day constitute a valid date.
- setYear(int year): It shall verify that the given year is between 1 and 9999.
- setMonth(int month): It shall verify that the given month is between 1 and 12.
- setDay(int day): It shall verify that the given day is between 1 and dayMax, where dayMax depends on the month and whether it is a leap year for Feb.
- getYear(), getMonth(), getDay(): return the value for the year, month and day, respectively.
- toString(): returns a date string in the format "xxxday d mmm yyyy", e.g., "Tuesday 14 Feb 2012".
- nextDay(): update this instance to the next day and return this instance. Take note that nextDay() for 31 Dec 2000 shall be 1 Jan 2001.
- nextMonth(): update this instance to the next month and return this instance. Take note that nextMonth() for 31 Oct 2012 shall be 30 Nov 2012.
- nextYear(): update this instance to the next year and return this instance. Take note that nextYear() for 29 Feb 2012 shall be 28 Feb 2013.
- previousDay(), previousMonth(), previousYear(): similar to the above.

Write the code for the MyDate class.

Use the following test statements to test the MyDate class:

```
MyDate d1 = new MyDate(2012, 2, 28);
System.out.println(d1);                 // Tuesday 28 Feb 2012
System.out.println(d1.nextDay());       // Wednesday 29 Feb 2012
```

```
System.out.println(d1.nextDay());      // Thursday 1 Mar 2012
System.out.println(d1.nextMonth()); // Sunday 1 Apr 2012
System.out.println(d1.nextYear());    // Monday 1 Apr 2013

MyDate d2 = new MyDate(2012, 1, 2);
System.out.println(d2);                          // Monday 2 Jan 2012
System.out.println(d2.previousDay());      // Sunday 1 Jan 2012
System.out.println(d2.previousDay());      // Saturday 31 Dec 2011
System.out.println(d2.previousMonth()); // Wednesday 30 Nov 2011
System.out.println(d2.previousYear());    // Tuesday 30 Nov 2010

MyDate d3 = new MyDate(2012, 2, 29);
System.out.println(d3.previousYear());    // Monday 28 Feb 2011

// MyDate d4 = new MyDate(2099, 11, 31); // Invalid year, month, or day!
// MyDate d5 = new MyDate(2011, 2, 29);    // Invalid year, month, or day!
```

Write a test program that tests the nextDay() in a loop, by printing the dates from 28 Dec 2011 to 2 Mar 2012.

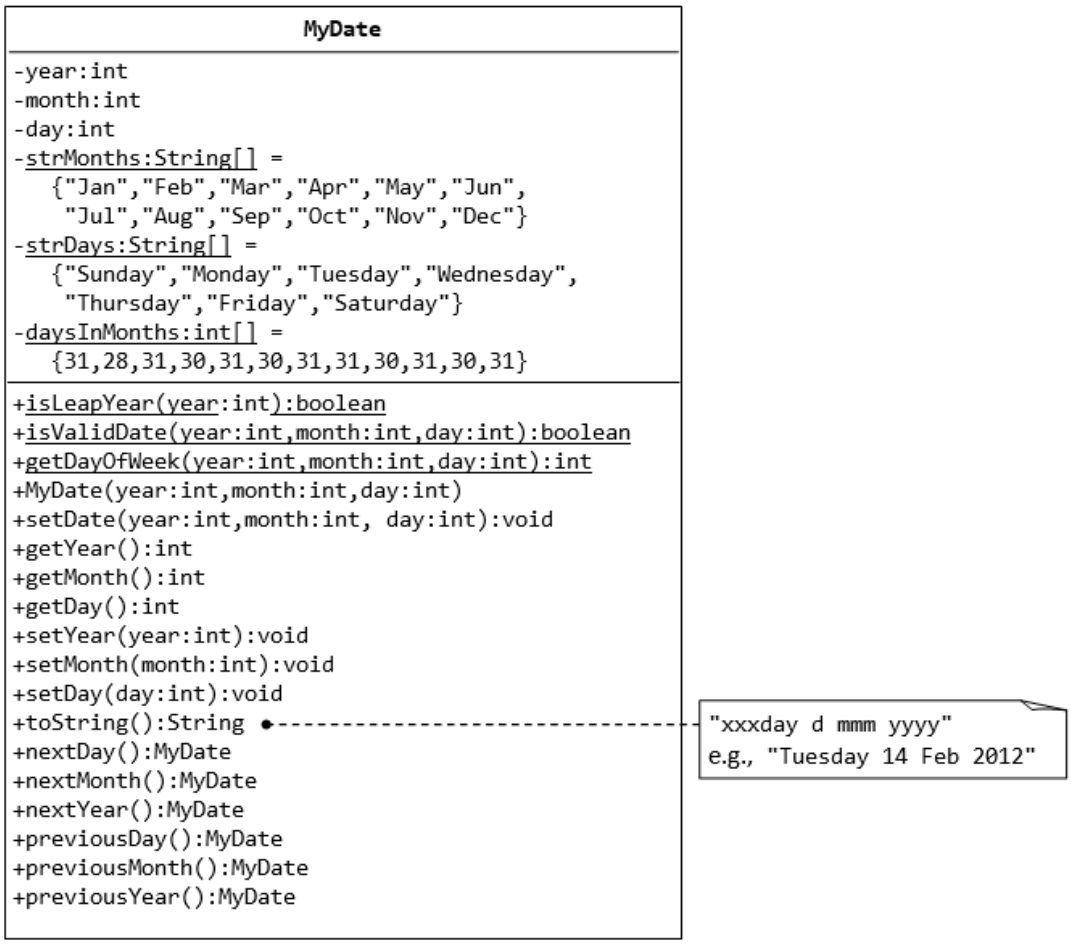

Figure 2: MyDate class diagram