# Exercise Two: Classes and Objects

**1. (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.

**You need to include the following main function in the submitted code to test the Book class:**

```
import java.util.Scanner;
public class Main{
    public static void main(String[] args) {
        Scanner x = new Scanner(System.in);
        int a = x.nextInt();
        int b = x.nextInt();
        int c = x.nextInt();
        MyInteger mi = new MyInteger(a);
        MyInteger ni = new MyInteger(b);
        System.out.println(mi.isEven());
        System.out.println(mi.isOdd());
        System.out.println(mi.isPrime());
        System.out.println(mi.isPrime(c));
        System.out.println(mi.isPrime(ni));
        System.out.println(mi.equals(c));
        System.out.println(mi.equals(ni));
        x.close();
    }
}
```

**input:**

```
    3 4 6
output:
    false
    true
    true
    false
    false
    false
    false:
```

**2. (Geometry: the Circle2D class) (30 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)).

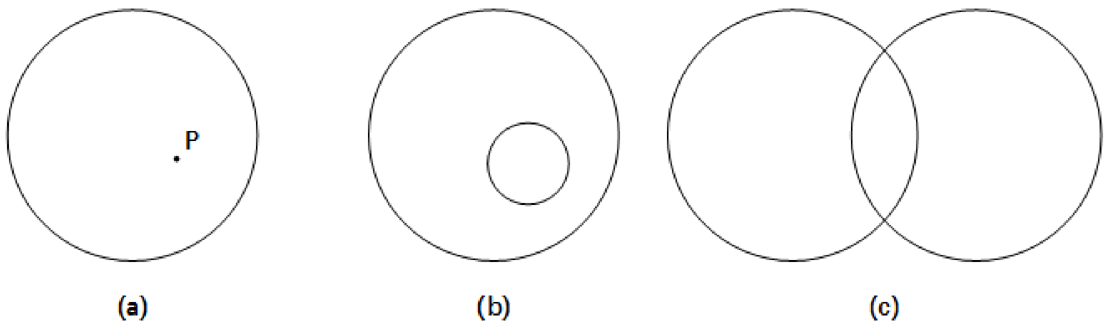
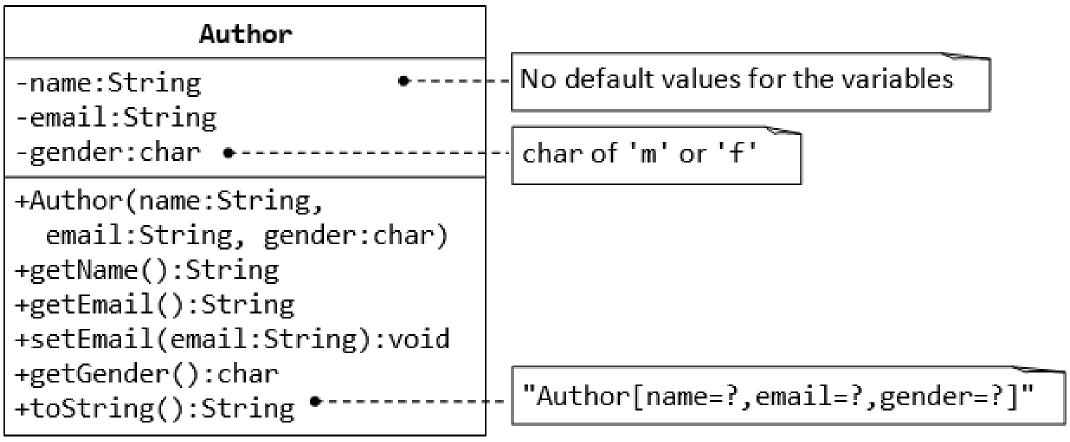


Figure 1: (a) A point is inside the circle. (b) A circle is inside another circle. (c) A circle overlaps another circle.

```
You need to include the following main function in the submitted code
to test the Book class:
```

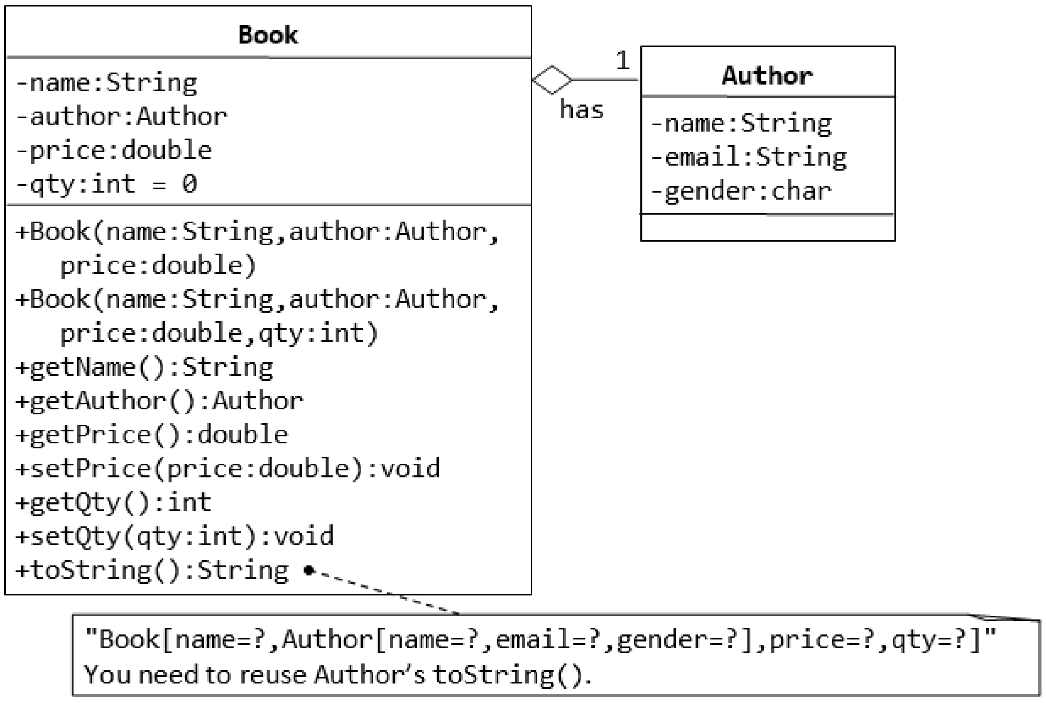
```
import java.util.Scanner;
public class Main {
     public static void main(String[] args) {
              Scanner input = new Scanner(System.in);
              Circle2D op1 = new
Circle2D(input.nextDouble(),input.nextDouble(),input.nextDouble());
              Circle2D op2 = new
Circle2D(input.nextDouble(),input.nextDouble(),input.nextDouble());
              double x = input.nextDouble();
              double y = input.nextDouble();

              System.out.println("The circle's area is "+op1.getArea());
              System.out.println("The circle's perimeter is
"+op1.getPerimeter());
              System.out.println("The circle overlaps with the specified
circle: "+op1.overlaps(op2));
              System.out.println("The circle contains the specified
point: "+op1.contains(x, y));
              System.out.println("The circle contains the specified
circle: "+op1.contains(op2));
              input.close();
     }

}
input:
    1 1 1 2 1 2 3 4
output:
    The circle's area is 3.141592653589793
    The circle's perimeter is 6.283185307179586
    The circle overlaps with the specified circle: true
    The circle contains the specified point: false
    The circle contains the specified circle: false
```

**3. (The Author and Book Classes) (40 Points)**



A class called Author (as shown in the class diagram) is designed to model a book's author. It contains:

- Three private instance variables: name (String), email (String), and gender (char of either 'm' or 'f');
- One constructor to initialize the name, email and gender with the given values;
  <u>public Author (String name, String email, char gender) {......}</u>
  (There is no default constructor for Author, as there are no defaults for name, email and gender.)
- public getters/setters: getName(), getEmail(), setEmail(), and getGender();
  (There are no setters for name and gender, as these attributes cannot be changed.)
- A toString() method that returns "Author[name=?,email=?,gender=?]", e.g., "Author[name=Tan Ah Teck,email=ahTeck@somewhere.com,gender=m]".



A class called Book is designed (as shown in the class diagram) to model a book written by one author. It contains:

- Four private instance variables: name (String), author (of the class Author you have just created, assume that a book has one and only one author), price (double), and qty (int);
- Two constructors:
  public Book (String name, Author author, double price, int qty) { ...... }
- public methods getName(), getAuthor(), getPrice(), setPrice(), getQty(), setQty().
- A toString() that returns "Book[name=?,Author[name=?,email=?,gender=?],price=?,qty=?". You should reuse Author's toString().

**You need to include the following main function in the submitted code to test the Book class:**

```
public class Main {
    public static void main(String[] args){
     Author ahTeck = new Author("Tan Ah Teck", "ahteck@nowhere.com",
'm');
     System.out.println(ahTeck);  // Author's toString()

     Book dummyBook = new Book("Java for dummy", ahTeck, 19.95, 99);  //
Test Book's Constructor
     System.out.println(dummyBook);  // Test Book's toString()

     // Test Getters and Setters
     dummyBook.setPrice(29.95);
     dummyBook.setQty(28);
     System.out.println("name is: " + dummyBook.getName());
     System.out.println("price is: " + dummyBook.getPrice());
     System.out.println("qty is: " + dummyBook.getQty());
     System.out.println("Author is: " + dummyBook.getAuthor());  //
Author's toString()
     System.out.println("Author's name is: " +
dummyBook.getAuthor().getName());
     System.out.println("Author's email is: " +
dummyBook.getAuthor().getEmail());

     // Use an anonymous instance of Author to construct a Book instance
     Book anotherBook = new Book("more Java", new Author("Paul Tan",
"paul@somewhere.com", 'm'), 29.95);
     System.out.println(anotherBook);  // toString()
    }
}
```

```
Output:
Author[name = Tan Ah Teck, email = ahteck@nowhere.com, gender = m]
Book[name = Java for dummy, Author[name = Tan Ah Teck, email =
ahteck@nowhere.com, gender = m], price = 19.95, qty = 99]
name is: Java for dummy
price is: 29.95
qty is: 28
Author is: Author[name = Tan Ah Teck, email = ahteck@nowhere.com,
gender = m]
Author's name is: Tan Ah Teck
Author's email is: ahteck@nowhere.com
Book[name = more Java, Author[name = Paul Tan, email =
paul@somewhere.com, gender = m], price = 29.95, qty = 0]
```

**4. (Stock Seller) (Bonus Question: 20 Points)**

Design a class named StockSeller, which can get the maximum profit under different constraints. It contains:

- Prices: An Integer Array, the *i<sup>th</sup>* number means the stock price of the *i<sup>th</sup>* day.
- StockSeller：The constructor, in which initiates the stock prices.
- MaxProfit1：Design an algorithm to find the maximum profit, only permitted to complete at most one transaction. Note that you cannot sell a stock before you buy.

```
Input: [7,1,5,3,6,4]

Output: 5

Explanation: Buy on day 2 (price = 1) and sell on day 5 (price = 6),
profit = 6-1 = 5.

Note: 7-1 = 6, as selling price needs to be larger than buying price.

Input: [7,6,4,3,1]

Output: 0

Explanation: In this case, no transaction is done, i.e. max profit = 0.
```

- MaxProfit2：Design an algorithm to find the maximum profit, you are permitted to buy and sell a stock multiple times. Note: You may not engage in multiple transactions at the same time (i.e., you must sell the stock before you buy again).

```
Input: [7,1,5,3,6,4]

Output: 7

Explanation: Buy on day 2 (price = 1) and sell on day 3 (price = 5),
profit = 5-1 = 4. Then buy on day 4 (price = 3) and sell on day 5
(price = 6), profit = 6-3 = 3.
```

```
Input: [1,2,3,4,5]

Output: 4

Explanation: Buy on day 1 (price = 1) and sell on day 5 (price = 5),
profit = 5-1 = 4.

Note:  You cannot buy on day 1, buy on day 2 and sell them later, as
you are engaging multiple transactions at the same time. You must sell
before buying again.
```

- MaxProfit3：Design an algorithm to find the maximum profit, you are permitted to complete at most two transactions. Note: You may not engage in multiple transactions at the same time (i.e., you must sell the stock before you buy again).

```
Input: [3,3,5,0,0,3,1,4]

Output: 6

Explanation: Buy on day 4 (price = 0) and sell on day 6 (price = 3),
profit = 3-0 = 3. Then buy on day 7 (price = 1) and sell on day 8
(price = 4), profit = 4-1 = 3.

Input: [1,2,3,4,5]

Output: 4

Explanation: Buy on day 1 (price = 1) and sell on day 5 (price = 5),
profit = 5-1 = 4. Note that you cannot buy on day 1, buy on day 2 and
sell them later, as you are engaging multiple transactions at the same
time. You must sell before buying again.
```

```
You need to include the following main function in the submitted code
to test the StockSeller class:
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
            Scanner x=new Scanner(System.in);
       while(x.hasNext()){
           int m=x.nextInt();
           int[] price = new int[m];
           for(int i=0;i<m;i++)
               price[i]=x.nextInt();
           StockSeller stock_seller = new StockSeller(price);
           System.out.println(stock_seller.MaxProfit1());
           System.out.println(stock_seller.MaxProfit2());
           System.out.println(stock_seller.MaxProfit3());
        }
        x.close();
     }
}

Output:
4
4
4
```