Q1. [Lab02-Q3]

Scanner object for file reading: `Scanner inFile = new Scanner(new File(filepathname));`

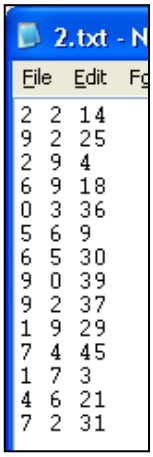
Rewrite the statement as guided below:

```
File f;
f = _____________________________;        //Create the File object
                                            (Note: the object variable refers to the object)

Scanner inFile;                           //Declare a variable for a Scanner object
inFile = _____________________________;   //Create the Scanner object to access the file
```

Q2. Refer to Lab02 Q4,

```
Please input the file pathname: c:\2.txt
  0   0   0  36   0   0   0   0   0
  0   0   0   0   0   0   0   3   0  29
  0   0  14   0   0   0   0   0   0   4
  0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0  21   0   0   0
  0   0   0   0   0   0   9   0   0   0
  0   0   0   0   0  30   0   0   0  18
  0   0  31   0  45   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0
 39   0  37   0   0   0   0   0   0   0
Maximum row sum: 76 (row 7,9)
Maximum col sum: 82 (col 2)
```



An approach is:

- Implement 2 small helper methods:

```
private int getRowSum(int r) //Return the sum of row r
private int getColSum(int c) //Return the sum of column c
```

- Then apply the above in

```
public int getRowSumMax() //return the maximum sum among the rows
public int getColSumMax() //return the maximum sum among the columns
public void printAllRowsOfMax() //print the row numbers which contain the max
public void printAllColsOfMax() //print the col numbers which contain the max
```

Your task: Complete the code below and on next page.

```
public static void main(String[] args) throws FileNotFoundException
{
    System.out.print("Please input the file pathname: ");
    Scanner scannerObj = new Scanner(System.in);
    String fileName = scannerObj.nextLine();

    Table2dMxSumRowCol table = new Table2dMxSumRowCol(fileName);
    table.print();

    System.out.print("Maximum row sum: "_____________________________);

    System.out.print(" (row ");

    _____________________________//Call table.printAllRowsOfMax();
    System.out.println(")");

    System.out.print("Maximum col sum: "_____________________________);

    System.out.print(" (col ");

    _____________________________//Call table.printAllColsOfMax();
    System.out.println(")");

    scannerObj.close();
}
```

```
public class Table2dMxSumRowCol {
  private int[][] nums;
  //.. some methods

  private int getRowSum(int r)
  {









  }

  public int getRowSumMax()
  {









  }

  public void printAllRowsOfMax()
  {

  }

  public int getColSumMax(){..} //Similar to getRowSumMax

  private int getColSum(int c){ .. } //Similar to getRowSum

  public void printAllColsOfMax(){..} //Similar to printAllRowsOfMax
```

Q3. Lab01 Q6 (OO Programming from C++ to Java)

Lab01.pdf (all), Given files
Q1-2 A Java class - Day Q1_Explain.pdf
Q3 Programming Graphics mode
Q4 OO sample Library Program
Q5 Day.previous
Q6 OO Programming from C++ to Java
C++ Problem Explained.pdf

(a) Mary is new to O-O programming. She has written a simple program that models shops (`class Shop`) and customers (`class Customer`), and how the shops earn profits from customers.

In the `main` function she adds 2 customer objects and 2 shop objects, sets the initial money amount of each customer, and calls the `earn` function of the shops to earn and get money from the customers. Finally she shows the customers' remaining total money and the shops' total profits.

Below are the source files written by Mary. Mary's expected program outputs are shown in the comments next to the output statements in the `main` function.

Main.cpp

```
int main()
{
    Customer c1,c2;
    Shop s1,s2;

    c1.set(200); //c1 has $200 initially
    c2.set(200); //c2 has $200 initially

    s1.earn(c1, 30); //s1 earns and get $30 from c1
    s2.earn(c1, 40); //s2 earns and get $40 from c1
    s2.earn(c2, 50); //s2 earns and get $50 from c2

    cout << c1.getAmount() << endl; //expected output: 130
    cout << c2.getAmount() << endl; //expected output: 150
    cout << s1.getProfit() << endl; //expected output: 30
    cout << s2.getProfit() << endl; //expected output: 90

    return 0;
}
```

Customer.h

```
class Customer {
private:
    int totalMoney;
public:
    void set(int money);
    void spend(int value);
    int getAmount();
};
```

Shop.h

```
class Shop {
private:
    int totalProfit = 0;
public:
    void earn(Customer c, int value);
    int getProfit();
};
```

Customer.cpp

```
void Customer::set(int money) {
    totalMoney = money;
}

void Customer::spend(int value) {
    totalMoney -= value;
}

int Customer::getAmount() {
    return totalMoney;
}
```

Shop.cpp

```
void Shop::earn(Customer c, int value) {
    totalProfit += value;
    c.spend(value);
}

int Shop::getProfit() {
    return totalProfit;
}
```

Mary can compile the source files and build the executable successfully. However, when she runs the program, the outputs are wrong. They are not the same as her expectations.

(b) Design the class `Group` that models a group of customers joined together for group-purchasing.

- For simplicity, we assume that a purchasing group has at most 10 customers only.
  Also, the money to pay for each group purchase is divisible by the count of group members so that the payment by each member is a whole number.
- The same customer should be able to purchase individually and should be able to join 1 or more purchasing groups. For example, if the customer has $100 initially, then after spending $40 individually, and $35 in one group purchase, and $12 in another group purchase, he should have $13 left.
- Other than the `Group` class, you also need to

  (a) Add an `earn` function in the `Shop` class that handles earning from group purchase

  (b) Rewrite the `main` function for testing. It should contain at least the following steps:
  - create 2 customer objects: `c1` and `c2`.
  - create 1 purchasing group `g1` and have both `c1` and `c2` joining `g1`
  - create 1 shop object: `s1`
  - the shop `s1` earns $300 from `g1` (i.e. each of `c1` and `c2` spends $150)
  - the shop `s1` earns $30 from `c1`
  - display the total profit of `s1`
  - display the total remaining money of `c1`

Main.java

```
public class Main{

    public static void main(String [] args)
    {
        Customer c1, c2;
        c1 = new Customer(200);
        c2 = new Customer(200);

        Group g1 = new Group();
        g1.add(c1);
        g1.add(c2);

        Shop s1 = new Shop();
        s1.earn(g1, 300); // s1 earns  totally $300 from the group
        s1.earn(c1, 30);
        System.out.println(s1.getProfit()); // output: 330
        System.out.println(c1.getAmount()); // output: 20
    }
}
```

Customer.java

```
public class Customer {

    private int totalMoney;

    public Customer(int money) {
        totalMoney = money;
    }

    public int getAmount() {
        return totalMoney;
    }

    public void spend(int value) {
        totalMoney -= value;
    }
}
```

Shop.java

```
public class Shop {
    private int totalProfit = 0;

    public int getProfit() {
        return totalProfit;
    }

    public void earn(Customer c, int value) {
        totalProfit += value;
        c.spend(value);
    }

}
```

Group.java