CIS 1068: Solution to Practice Problems 17

Practice for midterm 2: objects, files, methods, and algorithms.

1. Tracing programs

   a. What does the class below print to the screen?

   ```java
   public class MysteryClass
   {
       public static void main(String [] args)
       {
           mystery2();
           mystery1();
       }
       public static void mystery1() {
           System.out.println("hello!!!");
       }
       public static void mystery2() {
           System.out.println("goodbye!!!");
       }
   }
   
goodbye!!
hello!!
   
   b. What does the class below print to the screen?

   ```java
   public class MysteryClass2
   {
       public static void main(String [] args)
       {
           int x = 1;
           int y = 2;
           int z = mystery(y, x);
           System.out.println("x = " + x);
           System.out.println("y = " + y);
           System.out.println("z = " + z);
       }
       public static int mystery(int x, int y)
       {
           int temp = x + y;
           y = x * temp;
           x--;
           return temp;
       }
   }
   
x = 1
y = 2
z = 3
What does the class below print to the screen?

```java
import java.util.Arrays;
public class MysteryClass3 {
    public static void main(String [] args) {
        double [] arr1 = {3.5, 5.5, 7.5};
        double [] arr2 = {1.0, 3.0, 5.0};

        mystery(arr1, arr2);

        System.out.println(Arrays.toString(arr1));
        System.out.println(Arrays.toString(arr2));
    }

    public static void mystery(double [] a1, double [] a2) {
        for(int i=0; i<a1.length && i<a2.length; i++) {
            a1[i] = a1[i] + a2[i];
            if(i>0) {
                a2[i] -= a2[i-1];
            }
        }
    }
}
```

```
{4.5, 8.5, 12.5}
{1.0, 2.0, 3.0}
```
d. What does the class below print to the screen?

```java
public class MysteryClass4
{
    public static void main(String [] args)
    {
        Point [] p = { new Point(2, 4),
                       new Point(3, 5),
                       new Point(9, 11) };

        mystery1(p);
        System.out.println(Arrays.toString(p));
        p = mystery2(p);
        System.out.println(Arrays.toString(p));
    }

    public static void mystery1(Point [] ps)
    {
        for(int i=0; i<ps.length; i++) {
            ps[i].setLocation(ps[i].x + i, ps[i].y);
        }
    }

    public static Point [] mystery2(Point [] ps)
    {
        Point [] ret = new Point[ps.length];
        for(int i=0; i<ps.length; i++) {
            ps[i].translate(-1, 1);
        }
        return ret;
    }

    ${(2, 4), (4, 5), (11, 11)};
    {null, null, null};
```
e. Answer the questions below about the following class:

```java
public class MysteryClass5 {
    public static void main(String [] args) {
        Point p = new Point(3, 7);  // POINT 1
        Point p2 = new Point(5, 9);
        p2 = mystery(p, p2);
        // POINT 1
    }

    public static Point mystery(Point p1, Point p2) {
        Point ret = new Point(p1.x, p1.y);  // POINT 2
        p1.translate(p2.x, p2.y);
        p2.setLocation(-5, 5);  // POINT 2
        return p2;
    }
}
```

Question: What are the values of all variables at POINT 1 in main?

- **p** → (8, 16) (because it's a reference to the same place as p1 in mystery)
- **p2** → (3, 7) (because it is assigned the return value of mystery, which is a reference to the data that "ret" refers to)

Question: What are the values of all variables at POINT 2 in mystery?

- **ret** → (3, 7)
- **p1** → (8, 16)
- **p2** → (-5, 5)
What does the following code display on the screen?

```java
public class MysteryClass6 {

    public static void main(String [] args) {
        System.out.println("mystery2(true)" );
        mystery2(true); // 1
        System.out.println("mystery2(false)" );
        mystery2(false); // 2
        System.out.println("mystery1(true)" );
        mystery1(true); // 3
    }

    public static void mystery1(boolean b) {
        if(b) { // 4
            mystery2(!b); // 5
        }
        System.out.println("m1");
        System.out.println(b); // 6
    }

    public static void mystery2(boolean b) { // 7
        if(!b) { // 8
            mystery1(b); // 9
        }
        System.out.println("m2");
        System.out.println(b); // 10
    }
}
```

```plaintext
mystery2(true)
m2
true
mystery2(false)
m1
false
m2
false
mystery1(true)
m1
false
m2
false
m1
true
```
2. String and File processing
   a. Write a class to open a file called "decimal.txt" and read a single token from the file. Print the token to the screen.
      - Be careful to include all necessary "import" statements

   ```java
   import java.io.*;
   public class FileProcessor {
       public static void main(String [] args) throws FileNotFoundException {
           Scanner sc = new Scanner(new File("decimal.txt"));
           String next = sc.next();
           System.out.println(next);
           sc.close();
       }
   }
   ```

   b. How did you handle the checked exception, FileNotFoundException?
      - If you did not declare that your method "throws" this exception, and you did not surround your Scanner constructor with a try/catch combination, then your program won't compile.
      - Try writing the program both ways.
      - Question: what is the difference between a Checked exception and an Unchecked exception?

   // above, I handled the FNFException by declaring it to be thrown
   // here's the other way, try and catch:
   ```java
   import java.io.*;
   public class FileProcessor {
       public static void main(String [] args) {
           Scanner sc = null;
           try {
               sc = new Scanner(new File("decimal.txt"));
               String next = sc.next();
               System.out.println(next);
           }
           catch(FileNotFoundException e) {
               e.printStackTrace();
           }
           finally {
               if(sc!=null) {
                   sc.close();
               }
           }
       }
   }
   ```

   Answer to question: a Checked exception (like FileNotFoundException) requires that you either "catch" it or put in a declaration that says this method throws the exception. An Unchecked exception (like NullPointerException) doesn't require that you do either of those. Checked exceptions are exceptions that might happen through no fault of the programmers (like maybe somebody else came along and deleted the file you were trying to read in). Unchecked exceptions are always the programmer's fault.

   c. Modify your program so that it keeps reading and printing tokens to the screen until the file "decimal.txt" has no more tokens left.
import java.io.*;
public class FileProcessor {
    public static void main(String [] args) throws FileNotFoundException
    {
        Scanner sc = new Scanner(new File("decimal.txt");
        while(sc.hasNext()) {
            String next = sc.next();
            System.out.println(next);
        }
        sc.close();
    }
}

d. Modify your program so that it saves the tokens to an output file called "roman.txt", instead of printing them to the screen.

    import java.io.*;
    public class FileProcessor {
        public static void main(String [] args) throws FileNotFoundException
        {
            Scanner sc = new Scanner(new File("decimal.txt");
            PrintWriter pw = new PrintWriter(new File("roman.txt");
            while(sc.hasNext()) {
                String next = sc.next();
                pw.println(next);
            }
            sc.close();
            pw.close();
        }
    }

e. Remember to close your input and output files when your code is finished with them!

    f. Modify the loop in your program so that it only saves numbers, and not words, from the input file to the output file. It should just skip over any tokens that are not integers.

    import java.io.*;
    public class FileProcessor {
        public static void main(String [] args) throws FileNotFoundException
        {
            Scanner sc = new Scanner(new File("decimal.txt");
            PrintWriter pw = new PrintWriter(new File("roman.txt");
            while(sc.hasNext()) {
                if(sc.hasNextInt()) {
                    int nextInt = sc.nextInt();
                    pw.println(nextInt);
                } else {
                    String next = sc.next();
                    // skip printing these
                }
            }
            sc.close();
            pw.close();
        }
    }
g. **Modify your loop again so that for any numbers less than 1000, your code converts it into Roman numerals and saves those to the output file. Numbers \( \geq 1000 \) should be saved as regular integers. In Roman numerals, I = 1, V = 5, X = 10, L = 50, C = 100, D = 500, and M = 1000. To put them together, II = 2, IV = 4, VI = 6, VIII = 8, IX = 9, XI = 11, XIII = 13, XIV = 14, XIX = 19, XXI = 21, XL = 40, LX = 60, XC = 90, CX = 110, CD = 400, DC = 600, CM = 900.

```java
import java.io.*;
public class FileProcessor {
    public static void main(String[] args) throws FileNotFoundException {
        Scanner sc = new Scanner(new File("decimal.txt"));
        PrintWriter pw = new PrintWriter(new File("roman.txt"));
        while(sc.hasNext()) {
            if(sc.hasNextInt()) {
                int nextInt = sc.nextInt();
                if(nextInt >= 1000 || nextInt <= 0) {
                    pw.println(nextInt);
                } else {
                    String roman = "";
                    int hundredDigit = nextInt / 100;
                    if(hundredDigit == 9) {
                        roman += "CM";
                        hundredDigit = 0;
                    } else if(hundredDigit >= 5) {
                        roman += "D";
                        hundredDigit -= 5;
                    }
                    for(int i=0; i<hundredDigit; i++) {
                        roman += "C";
                    }
                    int tensDigit = nextInt / 10 % 10;
                    if(tensDigit == 9) {
                        roman += "XC";
                        tensDigit = 0;
                    } else if(tensDigit >= 5) {
                        roman += "L";
                        tensDigit -= 5;
                    }
                    for(int i=0; i<tensDigit; i++) {
                        roman += "X";
                    }
                    int onesDigit = nextInt % 10;
                    if(onesDigit == 9) {
                        roman += "IX";
                        onesDigit = 0;
                    } else if(onesDigit >= 5) {
                        roman += "V";
                        onesDigit -= 5;
                    }
                    for(int i=0; i<onesDigit; i++) {
                        roman += "I";
                    }
                    pw.println(roman);
                }
            } else {
                String next = sc.next();
            }
        }
    }
}
```
h. Modify the loop in your program so that it saves each line from the input file to a line in the output file (making a perfect copy of the input file).

```java
import java.io.*;
public class FileProcessor {
    public static void main(String[] args) throws FileNotFoundException {
        Scanner sc = new Scanner(new File("decimal.txt"));
        PrintWriter pw = new PrintWriter(new File("roman.txt"));
        while(sc.hasNextLine()) {
            String nextLine = sc.nextLine();
            pw.println(nextLine);
        }
        sc.close();
        pw.close();
    }
}
```

i. Modify the loop in your program so that it reverses each line from the input file, and saves the reversed line to the output file.

```java
import java.io.*;
public class FileProcessor {
    public static void main(String[] args) throws FileNotFoundException {
        Scanner sc = new Scanner(new File("decimal.txt"));
        PrintWriter pw = new PrintWriter(new File("roman.txt"));
        while(sc.hasNextLine()) {
            String nextLine = sc.nextLine();
            pw.println(nextLine.reverse());
        }
        sc.close();
        pw.close();
    }
}
```

j. Modify the loop in your program so that it only saves lines that are longer than 10 characters to the output file.

```java
import java.io.*;
public class FileProcessor {
    public static void main(String[] args) throws FileNotFoundException {
        Scanner sc = new Scanner(new File("decimal.txt"));
        PrintWriter pw = new PrintWriter(new File("roman.txt"));
        while(sc.hasNextLine()) {
            String nextLine = sc.nextLine();
            if(nextLine.length() > 10) {
                pw.println(nextLine);
            }
        }
        sc.close();
        pw.close();
    }
}
```
while(sc.hasNextLine()) {
    String nextLine = sc.nextLine();
    if(nextLine.length()>10) {
        pw.println(nextLine);
    }
}  
sc.close();
pw.close();

3. Primitive types and methods
   a. Fill out the class below so that it prints out 5 squares of different sizes on the screen, consisting of some ASCII character.

   public class SquarePrinter {
       public static void main(String [] args) {
           printSquare(10);
           printSquare(15);
           for(int i=4; i<16; i+=5) {
               printSquare(i);
           }
       }
       // your code here
       public static void printSquare(int width) {
           for(int i=0; i<width; i++) {
               for(int j=0; j<width; j++) {
                   System.out.print("*");
               }
               System.out.println();
           }
       }
   }

   b. Modify your printSquare method so that instead of printing the square to the screen, it saves the squares to a file called "square_output.txt".

   public static void printSquare(int width) {
       PrintWriter pw = new PrintWriter(new File("square_output.txt"));
       for(int i=0; i<width; i++) {
           for(int j=0; j<width; j++) {
               pw.print("*");
           }
           pw.println();
       }
   }
c. Change the printSquare method to a printRectangle method, where each rectangle's height might be different from the width. Modify the main method to call the printRectangle method instead of printSquare.

```java
public class SquarePrinter {
    public static void main(String [] args) {
        printRectangle(10, 12);
        printRectangle(15, 9);
        for(int i=4; i<16; i+=5) {
            printRectangle(i, i%7);
        }
    }

    public static void printRectangle(int width, int height) {
        for(int i=0; i<height; i++) {
            for(int j=0; j<width; j++) {
                System.out.print("*");
            }
            System.out.println();
        }
    }
}
```

d. Fill out the following class so that it generates an array containing two random integers between 0 and 99, and then computes the smallest number that is a multiple of both integers (the "least common multiple").

```java
import java.util.Random;

public class LeastCommonMultiple {
    public static void main(String [] args) {
        int [] nums = generateRandomArray(2, 100);
        int lcm = findLeastCommonMultiple(nums[0], nums[1]);
        System.out.println("least common multiple of ");
        System.out.println(nums[0]);
        System.out.println("and");
        System.out.println(nums[1]);
        System.out.println(" is: " + lcm);
    }

    public static int [] generateRandomArray(int len, int maxVal) {
        Random rand = new Random();
        int [] ret = new int[len];
        for(int i=0; i<ret.length; i++) {
            ret[i] = rand.nextInt(maxVal);
        }
        return ret;
    }
}
```
public static int leastCommonMultiple(int x1, int x2) {
    int lcm = x1;
    while(lcm%x2 != 0) {
        lcm += x1;
    }
    return lcm;
}

e. **Modify the findLeastCommonMultiple method so that it works for an array of ints (of any length > 0), rather than just 2 ints.

public static int leastCommonMultiple(int[] xs) {
    int lcm = 0;
    boolean isMultipleOfEach = true;
    do {
        lcm += xs[0];
        isMultipleOfEach = true;
        for(int i=1; i<xs.length; i++) {
            if(lcm % xs[i] != 0) {
                isMultipleOfEach = false;
                break;
            }
        }
    } while(!isMultipleOfEach);
    return lcm;
}

f. Modify the main method so that it generates an array of 10 random numbers between 0 and 199, and finds the least common multiple of all of them.

public static void main(String[] args) {
    int[] nums = generateRandomArray(10, 200);
    int lcm = findLeastCommonMultiple(nums);
    System.out.println("least common multiple of ");
    System.out.println(Arrays.toString(nums));
    System.out.println(" is: " + lcm);
}
4. Object methods
   
a. Modify the "generateRandomArray" method from above so that it creates an array of random Point objects instead of an array of ints. The x and y coordinates of each Point should be between 0 and some maximum, specified as a parameter.

   ```java
   public static Point[] generateRandomArray(int len, int maxVal)
   {
       Random rand = new Random();
       Point[] ret = new Point[len];
       for(int i=0; i<ret.length; i++) {
           int x = rand.nextInt(maxVal);
           int y = rand.nextInt(maxVal);
           ret[i] = new Point(x, y);
       }
       return ret;
   }
   ```

   b. Write a new class with a main method that calls this new "generateRandomArray" method.

   ```java
   import java.util.Random;
   public class PointArray
   {
       public static void main(String[] args)
       {
           Point[] ps = generateRandomArray(20, 100);
       }

       public static Point[] generateRandomArray(int len, int maxVal)
       {
           Random rand = new Random();
           Point[] ret = new Point[len];
           for(int i=0; i<ret.length; i++) {
               int x = rand.nextInt(maxVal);
               int y = rand.nextInt(maxVal);
               ret[i] = new Point(x, y);
           }
           return ret;
       }
   }
   ```

c. Add a method to your class that shifts the coordinates of every Point in a Point [] by +2 in the y coordinate.

   ```java
   public static void shiftYBy2(Point[] ps)
   {
       for(int i=0; i<ps.length; i++) {
           ps[i].translate(0, 2);
       }
   }
   ```

d. Add a statement to your main method that calls this method.

   ```java
   public static void main(String[] args)
   {
   ```
Point [] ps = generateRandomArray(20, 100);
    shiftBy2(ps);
}

e. When the class runs the statement in the main method to shift the y coordinates, does it modify the original points, or only a copy of the points in the method? Why?

The statement in the main method modifies the original points, since the copy in the shiftBy2 method is a reference to the same data (same set of Points).

f. Complete the class below so that when executed, it computes a Point that is in the center of the 3 Points that the user enters. The center Point has x and y coordinates that are the average of the other 3 Points. The program should display the center Point, then update the location of the three other Points so that they are centered on (0,0). It should then print the value of three Points in their new locations.

```java
import java.util.Scanner;
import java.util.Arrays;

public class PointMover {

    public static void main(String [] args) {
        Point [] points = read3Points();
        Point center = findCenterPoint(points);
        System.out.println("center is: " + center);
        movePoints(points, center);
        System.out.println("points in new location: " + Arrays.toString(points));
    }

    public static Point [] read3Points() {
        Scanner kb = new Scanner(System.in);
        Point [] ret = new Point[3];
        for(int i=0; i<ret.length; i++) {
            System.out.println("please enter x coord:");
            int x = kb.nextInt();
            System.out.println("please enter y coord:");
            int y = kb.nextInt();
            ret[i] = new Point(x, y);
        }
        return ret;
    }

    public static Point findCenterPoint(Point [] points) {
        Point ret = new Point();
        for(int i=0; i<points.length; i++) {
            ret.x += points[i].x;
            ret.y += points[i].y;
        }
        return ret;
    }
}
```
ret.x = ret.x / points.length;
ret.y = ret.y / points.length;
return ret;
}

class MovePoints {
  public static void movePoints(
    Point [] points,
    Point toMove)
  {
    for(int i=0; i<points.length; i++)
    {
      points[i].translate(-toMove.x,-toMove.y);
    }
  }
}