CIS1068 Practice Problems: Object & methods

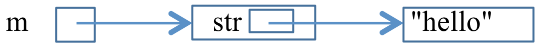
1. Tracing programs

a. What gets printed to the screen when we execute the MysteryClient class on the left?

```
public class MysteryClient
{
  public static void main(
                    String [] args)
  {
    Mystery m = new Mystery("hello");
    m.display(" again");
  }
}
```

```
public class Mystery
{
  public String str = null;
  public Mystery(String s)
  {   str = s; }

  public void display(String s)
  {   System.out.println(str + s); }
}
```

The first line in main() creates an object of type Mystery, and a reference variable pointing to it called m. It does not actually display anything, but here is what is in memory after that instruction:



Then the second line in main() executes, which calls the display() method in the Mystery class using object m. This method displays the concatenation of str and the parameter s, which the calling method has set to " again". So the output is:

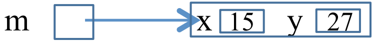
```
hello again
```

b. What gets printed to the screen when we execute the MysteryClient class on the left?

```
public class MysteryClient
{
  public static void main(
                    String [] args)
  {
    Mystery m = new Mystery(15, 27);
    m.display(3);

    Mystery m2 = new Mystery(16, 28);
    m2.reduce(4);
    m2.display(4);
  }
}
```

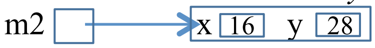
```
public class Mystery {
  public int x = 0;
  public int y = 0;

  public Mystery(int num1, int num2)
  {
    x = num1;
    y = num2;
  }

  public void display(int z)
  {
    if(x%z == 0 && y%z == 0) {
      System.out.println("divides");
    } else {
      System.out.println("too bad");
    }
  }

  public void reduce(int z)
  {
    x = x / z;
    y = y / z;
  }
}
```

The first line creates an object of type Mystery, and a reference to it called m. This doesn't display anything, but here's what's in memory after that instruction:

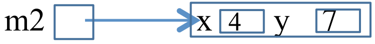


The next line passes a value of 3 to the parameter z in the display method. The display method tests whether 3 divides 15 and 27 (which is true), so it prints out: **divides**

The next line creates another object of type Mystery, and a reference to it called m2. Here's what that creates in memory:



The next line calls the reduce method, and passes a value of 4 to the z parameter. The reduce method sets the x field of m2 to its previous x field divided by z, and then sets the y field to its previous y field divided by z. Here's what m2 looks like in memory afterwards:



The last line in main calls the display method, and passes a value of 4 to the z parameter. The display method checks whether z divides m2's x attribute (which is true) and whether z divides m2's y value (which is false). So it displays: **too bad**

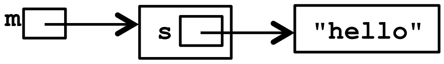
The display result is:

```
divides
too bad
```

c. What are the values of the variables in main() at POINT 1?

```
public class MysteryClient
{
  public static void main(
                    String [] args)
  {
    Mystery m = new Mystery("hello");
    // POINT 1
  }
}
```

```
public class Mystery
{
  public String str = null;
  public Mystery(String s)
  {   str = s;  }
}
```

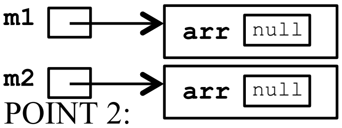


d. What are the values of the variables in main() at POINT 1? and at POINT 2? and 3?

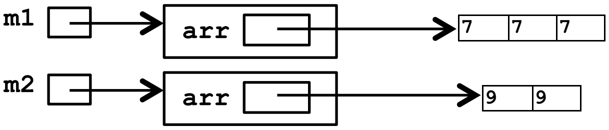
```
public class MysteryClient
{
  public static void main(
                    String [] args)
  {
    Mystery m1 = new Mystery();
    Mystery m2 = new Mystery();
    // POINT 1

    m1.setArr(3, 7);
    m2.setArr(2, 9);
    // POINT 2

    int x = m1.getVal(2);
    int y = m1.getVal(2);
    int z = m2.getVal(0);
    // POINT 3
  }
}
```

```
public class Mystery
{
  public int [] arr = null;

  public void setArr(
                int len, int val)
  {
    arr = new int[len];
    for(int i=0; i<len; i++) {
      arr[i] = val;
    }
  }

  public int getVal(int pos) {
    arr[pos]++;
    return arr[pos];
  }
}
```

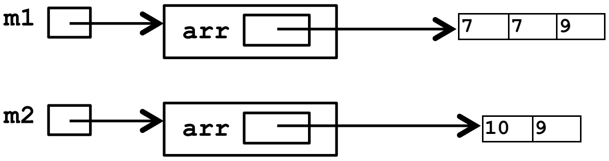
POINT 1:



POINT 2:



POINT 3:



x = 8
y = 9
z = 10

2. Object methods

a. Write a class Account with a constructor that accepts a series of charge accounts as its argument. These numbers should be stored in an array *records* that is initiated in the class as a **private** attribute. Then, write an accessor that accepts an account number as its argument. If this test account number is in the array *records*, true should be returned, otherwise, false.

```java
package account;
import java.util.Scanner;

public class Account {
    private int records[];

    public Account(int [] acc){
        records = new int [acc.length];

        for(int i = 0; i< acc.length; i++)
         records[i]=acc[i];
    }

    public boolean valid(int v){
        for(int i=0; i<records.length; i++)
            if(records[i] == v)
                return true;
        return false;
    }
}

```

```java
package account;
import java.util.Scanner;

public class AccountApplication {

   public static void main(String[] args) {

        int[] data = {5658845,4250125,7895122,8777541,
                      8451277,1302850,8080152,4562555,
                      5552012,5050552,7825877,1250255,
                      1005231,6545231,3852085,7576651,
                      7881200,4581002};

        Account a = new Account(data);

        Scanner kb = new Scanner (System.in);
        System.out.println("Input your account number:");

        int acc = kb.nextInt();
        System.out.println("The account is valid? "
               +a.valid(acc));
    }
}

```

b. Write a class DirverExam that has a constructor that accepts an array of char as the correct answers of the local driver's license exam. The records must be saved in a private attribute *key*. The class must have another **private** array *answers* to save the student's answers. The class contains a method *testing* to allow this student to input all answers (by nextLine().toUpperCase().charAt(0)) to match the *key*. When the student cannot ensure the answer, he/she can key in <enter> directly. See the demo of this testing in the below. This class also should have the following methods (public):

- Passed ( ), returns true if the student passed the exam, or false if the student failed. The exam has 20 multiple questions. A student must correctly answer 15 of the 20 questions to pass the exam.
- totalCorrect ( ), returns the total number of correctly answered questions.
- totalIncorrect ( ), returns the total number of incorrectly answered questions.
- questionsMissed ( ), returns an array of integers that contains the numbers of the questions that the student missed or answered incorrectly. Note that in the student score array, any character other than ‘A’, ‘B’, ‘C’, or ‘D’ will also be treated as an answer missed.

This class should support another class DL with the only static main method. That main method can test all the above methods.

```
package dl;

import java.util.Scanner;

public class DriversExam {
    private char[] key;
    private char[] answers;

    public DriversExam(char [] k){
        key = new char[k.length];
        for(int i=0; i<k.length; i++) {
           key[i]=k[i];
        }
    }
    public void testing(){
        Scanner input = new Scanner (System.in);
        System.out.println("--- Driver's testing ---\n");
        answers = new char[key.length];

        for(int i=0; i<key.length; i++){
            System.out.print("Question #" + (i+1) + " ANS> ");
            String str = input.nextLine().toUpperCase();

            answers[i] = (str.length()>0)? str.charAt(0):' ';
        }
    }

    public boolean passed(){
        if(totalCorrect() >= 15)
            return true;
        return false;
    }

    public int totalCorrect(){
        int count = 0;
        for(int i=0; i<answers.length; i++)
            if(answers[i] == key[i])
                count++;
        return count;
    }

    public int totalIncorrect(){
        return key.length - totalCorrect();
    }
```

```java

    public int[] questionsMissed(){
        int[] missed = new int[totalIncorrect()];
        int index = 0;
        for(int i=0; i<answers.length; i++)
            if(answers[i] != key[i]){
                missed[index] = i+1;
                index++;
            }
        return missed;
    }
}
```

```java
package dl;

import java.util.Arrays;

public class DL {
    public static void main(String[] args) {
        char[] key  =    {'B','D','A','A','C',
                          'A','B','A','C','D',
                          'B','C','D','A','D',
                          'C','C','B','D','A'};

        DriversExam d = new DriversExam(key);

        d.testing();

        System.out.println("Has this student passed the test? "+d.passed());
        System.out.println("The number of questions correctly answered: "+d.totalCorrect());
        System.out.println("The total number of questions incorrectly answered: "+d.totalIncorrect());
        System.out.println("The quesitons that this student missed are: "+Arrays.toString(d.questionsMissed()));
    }
}

```

c. Write a class GradeBook with a constructor that accepts five students' names, grades, and 4 test scores and save in the **private** fields:

- names, a string array to hold five students' names
- grades, an array of 5 characters to hold the five students' letter grades.
- testScores, a 2-dimentional array to hold 4 test scores for each of five students.

The class should have a constructor for the user to enter each student's name and his or her four test scores. The constructor will use an associate method (private double getAvg(int index)) to obtain the average test score, and then convert to a letter grade. This method will use the following **Numeric-To-Letter-Grade Scale:** 90-100 A, 80-89 B, 70-79 C, 60-69 D, 0-59 F. The accessor (public String toString) will return a string that carries all names, their average scores, and letter grades.

This class should support another class Grade with the only static main method. That main method can test all the above accessors. Note that any score less than 0 or greater 100 is not acceptable.

```java
package grade;
import java.util.Scanner;

public class GradeBook {
    private String[] names ;
    private char[] grades ;
    private int [][] testScores ;

    public GradeBook(int n){
        names = new String[n];
        grades = new char [n];
        testScores = new int [n][4];

        Scanner input = new Scanner (System.in);

        System.out.println("--- Grade Book ---\n");
        int i=0;
        for(; i<names.length; i++){
            System.out.print("Enter Student "+(i+1)+ " Name: ");
            names[i] = input.nextLine();
        }

        for(i=0; i<names.length; i++){
            System.out.println("Enter scores for " + names[i] + ": ");
            for(int c=0; c<testScores[0].length; c++){
                System.out.print("     Test #" + (c+1) + "(Score 0-100): ");
                do{
                testScores[i][c] = input.nextInt();
                }while(testScores[i][c] < 0 || testScores[i][c] > 100);
            }
            setGrade(i, getAvg(i));
        }
    }

    private double getAvg(int index){
        int i=0;
        double sum=0;
        for(; i<testScores[0].length; i++){
            sum += testScores[index][i];
        }
        return sum/i;
    }

    private void setGrade(int index, double g){
        if(g>=90)
             grades[index]= 'A';
        else if(g>79)
             grades[index]=  'B';
        else if(g>69)
             grades[index]=  'C';
        else if(g>59)
             grades[index]=  'D';
        else
```

```
             grades[index]=  'F';
    }

    public String toString(){
      int i;
      String result ="Names\t"+names[0];
      for(i = 1; i< names.length; i++){
         result += "\t"+names[i];
      }
      result += "\nAvg\t"+getAvg(0);
      for(i = 1; i< names.length; i++){
         result += "\t"+getAvg(i);
      }
      result += "\nFinal\t"+grades[0];
      for(i = 1; i< names.length; i++){
         result += "\t"+grades[i];
      }
      return result+"\n";
   }
}


```

```
package grade;
public class Grade {
    public static void main(String[] args) {
        GradeBook g = new GradeBook(5);
        System.out.println(" " + g);
    }
}

```