German University in Cairo
Media Engineering and Technology Faculty
Prof. Dr. Slim Abdennadher

June 6, 2015

CSEN202: Introduction to Computer Programming
Spring Semester 2015
Final Exam

Bar Code

Instructions: Read carefully before proceeding.

1) Duration of the exam: 3 hours (180 minutes).
2) No books or other aids are permitted for this test.
3) This exam booklet contains 16 pages, including this one. Three extra sheets of scratch paper are attached and have to be kept attached. **Note that if one or more pages are missing, you will lose their points. Thus, you must check that your exam booklet is complete**.
4) Write your solutions in the space provided. If you need more space, write on the back of the sheet containing the problem or on the four extra sheets and make an arrow indicating that. **Scratch sheets will not be graded unless an arrow on the problem page indicates that the solution extends to the scratch sheets**.
5) When you are told that time is up, stop working on the test.

Good Luck!

Don't write anything below ;-)

Exercise	1	2	3	4	5	6	$\sum$
Possible Marks	17	12	12	12	12	25	90
Final Marks							

Exercise 1 (6+3+8=17 Marks)

a) Given the following Java class:

```
public class MethodTester1
{
    public static int balabizo(int i)
    {
        int j = i + 3;
        return j;
    }

    public static void chico(int[] b, int j)
    {
        int i = j + 1;
        b[balabizo(i)] = b[j];
    }

    public static void main(String[] args)
    {
        int[] a = { 1, 1, 2, 3, 5, 8, 13};
        int i = 1;
        chico(a, i);
        System.out.println("Slim says " + balabizo(i));
        System.out.println("Rimon says " + a[balabizo(i)]);
        System.out.println("Wael says " + a[balabizo(i+1)]);
    }
}
```

What does the program display? Justify your answer.

Solution:

```
Slim says 4
Rimon says 5
Wael says 1
```

b) Given the following Java class:

```
public class MethodTester2 {

   public static void methodB(int[] c, int d) {
       c[0]++;
       d += 42;
   }

   public static int methodA(int[] a, int b) {
       methodB(a, b);
       a[0]++;
       return b/2;
   }

   public static void main(String[] args) {
       int[] arr = {8, 9, 10};
       int x = 1;
       x = methodA(arr, x);
       System.out.println(arr[0] + " " + x);
   }
}
```

What does the program display? Justify your answer.

Solution:

```
10 0
```

c) Given the following class:

```
public class IntObj
{
  int v;

  public void setInt(int v)
  {
    this.v = v;
  }

  public int getInt()
  {
    return v;
  }

public static void swap(IntObj o1, IntObj o2)
{
    IntObj o3 = o1;
    o1 = o2;
    o2 = o3;
}


  public static void main(String[] args)
  {
     IntObj p;
     p = new IntObj();
     IntObj q = new IntObj();
     q.setInt(20);
     p.setInt(q.getInt());           //line 1
     p.setInt(15);
     swap(p,q);                      //line 2
     q = p;                          //line 3
     p = null;
     System.out.print(p.getInt());   //line 4
  }
}
```

- What are the values stored in the locations that `p` and `q` reference after the line labeled `//line 1` is executed? Justify your answer.

  Solution:

  ```
  Value in p is 20
  Value in q is 20
  ```

- What are the values stored in the locations that `p` and `q` reference after the line labeled `//line 2` is executed? Justify your answer.

  Solution:

  ```
  Value in p is 15
  Value in q is 20
  ```

- What are the values stored in the locations that `p` and `q` reference after the line labeled `//line 3` is executed? Justify your answer.

  Solution:

```
Value in p is 15
Value in q is 15
```

- What occurs when `//line 4` is executed? Justify your answer.

  Solution:

  ```
  Exception in thread "main" java.lang.NullPointerException
  at IntObj.main(IntObj.java:48)
  ```

Exercise 2 (12 Marks)

The prime factors of a number are all of the prime numbers that will exactly divide the given number.

For example,

- Prime factors of 35 are 7 and 5, i.e. $35 = 5 \times 7$.
- Prime factors of 63 are 3, 3, and 7, i.e. $63 = 3 \times 3 \times 7$.
- Prime factors of 78 are 2, 3 and 13, i.e. $78 = 2 \times 3 \times 13$.

Your task is to write a **recursive method** to find the prime factors of a given number.

Complete the following code that given a number displays the factors of this number.

For input `63` the program should output

```
 3 3 7
```

```
public static void main(String[] args) {
     factorizeRec(Integer.parseInt(args[0]));
}


public static void factorizeRec(int n)
{
```

Solution:

```
public static void factorizeRec(int n)
    helperFactorize(2, n);
    System.out.println();
}

public static void helperFactorize(int a, int b) {
    if(a <= b) {
        if(b%a == 0) {
            System.out.print(" " + a);
            helperFactorize(a, b/a);
    }
    else {
        a = a+1;
        helperFactorize(a, b);
    }
}
```

Exercise 3 (12 Marks)

Write an iterative method `absoluteDifference` that takes as input an array of floating-point numbers and returns the smallest absolute difference between any pair of numbers in the array.

For example,

- for an input array

  ```
  double[] a = { 4.5, 3.5, 6.0, 20.0, 3.0 }
  ```

  the method should return `0.5` which corresponds to `3.5 - 3.0`.

- for an input array

  ```
  double[] a = { 8.2, 3.5, 6.0, 3.5, 20.0}
  ```

  the method should return `0.0` which corresponds to `3.5 - 3.5`.

Hint: You could use the method `Math.abs(double n)` that calculates the absolute value of a floating-point number `n`.

Solution:

```
public static double absoluteDifference(int[] a) {
    int N = a.length;
    double min = Double.POSITIVE_INFINITY;
    for (int i = 0; i < N; i++)
        for (int j = i+1; j < N; j++)
        {
            double delta = Math.abs(a[i] - a[j]);
            if (delta < min)
            min = delta;
        }
        return min;
}
```

Exercise 4 (12 Marks)

Write a recursive method `mergeRec` that given two array of integers displays the elements of the given arrays in an alternating way. Note that the two arrays could be of different length.

Note that you are not allowed to use any additional arrays.

Once you execute the following `main` method

```
public static void main(String[] args) {

    int[] a = {1,8,3,4};
    int[] b = {5,2};
    mergeRec(a,b);

}
```

the following should be displayed:

```
1 5 8 2 3 4
```

```
public static void mergeRec(int[] a, int[] b) {
```

Solution:

```
public static void mergeRec(int[]a, int[]b)
{
      helper(a,b,0,0);
}

public static void helper (int[]a, int[]b, int i, int j)
{
    if(a.length==i && b.length==j)
    {
       return;
    }

    if(b.length==j)
    {
      System.out.print(a[i]+" ");
      helper (a,b,++i,j);
    }
    else{
       if(a.length==i)
       {
         System.out.print(b[j] + " ");
         helper (a,b,i,++j);
       }
       else
       {
          System.out.print(a[i] + " " + b[j] +" ");
          helper(a,b,++i,++j);
       }
    }
  }
```

Exercise 5 (12 Marks)

Write a method `flatten` that, given a two-dimensional array of numbers, returns the flattened one-dimensional array that consists of all elements that occur in the input array.

The method should work correctly for ragged and non-ragged arrays.

Example: For the following two-dimensional array

```
1   3   17
51  11   4   6
1   3
4   2    3   6    9
```

your program should return the following one-dimensional array:

```
1   3   17  51  11   4   6  1   3  4   2   3   6    9
```

Solution:

```
public static int[] flatten(int[][] x) {

    int c = 0;

   for (int i = 0; i < x.length; i++) {
      for (int j = 0; j < x[i].length; j++) {
          c++;
      }
    }

    int [] a= new int[c];
    int k = 0;

    for (int i = 0; i < x.length; i++) {
       for (int j = 0; j < x[i].length; j++) {
          a[k++] = x[i][j];
       }
     }
     return a;

}
```

Exercise 6 (25 Marks)

The goal of this exercise is to design and implement a simple to-do list class.

a) Write a class `TodoItem` that represents an item in a to-do list. It should have the following attributes:

- a `String` that holds a description of the item that needs to be done (ex. `"walk dog"`)
- an `int` that holds the month (converted to a number) that the item is due
- an `int` that holds the day that the item is due
- a `boolean` variable `isDone` that holds whether or not the item has been completed.

The class should also have the following static class variables:

- an `int numItems` that holds how many `TodoItem` instances have been made
- an `int numDone` that holds how many `TodoItem` instances have been completed

Write the following constructors/methods for your class:

- a constructor that takes in the description of the to-do item, the day it is due, and the month it is due (as an `int`). This constructor should initialize the boolean variable `isDone` to `false`.
- a second constructor that only takes in the description of the to-do item. This constructor should initialize the boolean variable `isDone` to `false`.
- a `toString()` to display all information about the to-do item.
- a `itemDone` method that checks off an item as being done. It should update the corresponding variables accordingly.

Solution:

```
public class TodoItem {

  String description;
  int month;
  int day;
  boolean isDone;

  static int numItems=0;
  static int numDone=0;

  public TodoItem(String desc, int m , int d) {
      description = desc;
      month = m;
      day = d;
      numItems++;
  }

  public TodoItem(String description){
      this.description = description;
      numItems++;
   }

   public String toString(){

        return "Description: "+ description + " Month: "+ month +
         " Day: "+ day +" Done : "+isDone;
   }
```

```
    public void itemDone(){
        this.isDone=true;
        numDone++;
    }
}
```

b) Write a class `TodoList` that represents a to-do list. It will use the `TodoItem` class.

The `TodoList` class should contain two instance variables:

- an array of `TodoItem` instances, that represents the to-do list.
- an `int numItems` that holds the number of `TodoItem` instances that are actually in the array (the remaining slots will be null).

Write the following constructors/methods for this class:

- a constructor that takes an array `a` of `TodoItem` objects and initializes the corresponding instance variables. The array instance variable should be of the same length of `a`. Assume that the first `numItems` of `a` are `TodoItem` instances and the remaining elements are null. You are required to do a deep copy of the objects. Do not copy references of the objects. **Hint:** You are required to determine the value of `numItems`.
- an `add` method that should take in a `TodoItem` as a parameter, and add it to the `TodoItem` array in the first empty slot. The variable `numItems` should be updated accordingly. If the array is already full of `TodoItem` instances, this method should print a message saying so.
- a `delete` method that should take in an `int`, and return the `TodoItem` at that index in the `TodoItem` array. If there is no `TodoItem` instance at that index, this method should return null. Addtionally, all `TodoItem` instances in the array at higher indices should be moved one position lower to delete this `TodoItem` from the array. For example, If we delete the `TodoItem` instance at position 5, we would save it to be returned at the end of the method, and then move the `TodoItem` in position 6 into position 5, and then the `TodoItem` in position 7 into position 6, etc. Finally, update the instance variable `numItems`.
- a `toString` that should display the `TodoItem` instances in the array as a to-do list. Make sure that only `numItems` objects are displayed.
- a `main` that should do the following:
  - Create an array `items` of `TodoItem` objects of size 10. The list should consist of two `TodoItem` instances.
  - Create a `TodoList` instance using the array `items` and the corresponding constructor.
  - Add to the `TodoList` an additional `TodoItem`.
  - Delete the fifth element in the `TodoList`.
  - Display the information of the `TodoList` object.

Solution:

```
public class TodoList {

TodoItem[] items;
int numItems;

    public TodoList(TodoItem[] n) {
     items = new TodoItem[n.length];
     for(int i =0;i<n.length &&n[i]!=null; i++){
          items[i]= new TodoItem(n[i].description, n[i].month, n[i].day);
          numItems++;
```

```
    }
   }

   public void add(TodoItem item){
    if(numItems < items.length)
    items[numItems++]= item;
    else
    System.out.println("You can't add any more items
                   to your list since it is full.");
   }

   public TodoItem delete(int index){
    if(items[index]==null)
    return null;

    TodoItem deletedItem = items[index];

    for(int i =index; i<numItems; i++){
        items[i]=items[i+1];
    }
    numItems--;
    return deletedItem;

   }

   public  String toString(){
    String list ="";
    int j;
    for(int i=0; i<numItems; i++){
       j =i+1;
       list+= "Item "+ j+": " +items[i]+"\n";
    }
    return list;

   }

   public static void main(String[]args){
    TodoItem[] items = new TodoItem[10];
    items[0]= new TodoItem("Study for the CSEN202 final", 6,4);
    items[1]= new TodoItem("Study for the physcis final", 6,6);
    TodoList list = new TodoList(items);
    list.add(new TodoItem("Work out"));
    System.out.println(list.delete(4));
    System.out.println(list);
   }


}
```

Scratch paper

Scratch paper

Scratch paper