COE 212 – Engineering Programming

Welcome to Exam II
Thursday April 30, 2015

Instructors: Dr. Georges Sakr
Dr. Joe Tekli
Dr. Wissam F. Fawaz

Name: _____

Student ID: _____

Instructions:

1. This exam is **Closed Book**. Please do not forget to write your name and ID on the first page.
2. You have exactly **115 minutes** to complete the 5 required problems.
3. Read each problem carefully. If something appears ambiguous, please write your assumptions.
4. Do not get bogged-down on any one problem, you will have to work fast to complete this exam.
5. Put your answers in the space provided only. No other spaces will be graded or even looked at.

Good Luck!!

Problem 1: Multiple choice questions (**15 minutes)** [14 points]

Consider an input text file called "**in.txt**" that contains some data and an empty output file named "**out.txt**".

1) Which of the following statements creates an object called `inFromFile` that can obtain data from **in.txt**?
   a. `Scanner fileScan = new Scanner(new File("in.txt"));`
   b. `Scanner inFromFile = new Scanner(new File("out.txt"));`
   c. **`Scanner inFromFile=new Scanner(new File(new String("in.txt"));`**
   d. None of the above
2) Given the `inFromFile` object created earlier, which of the following code fragments prints to the screen all the lines contained in **in.txt** each on a different line?
   a.
```
while(inFromFile.hasNext()) {
        System.out.println(inFromFile.nextLine());};
```
   b.
```
while(inFromFile.hasNext()) {
        System.out.println(inFromFile.nextLine());}
```
   c. Both of the above
   d. None of the above
3) Which of the following creates an object that can be used to parse the pieces of a line obtained from **in.txt**? Note that we are interested in the tokens separated by the / (**forward slash**) character.
   a.
```
Scanner lineScan = new Scanner(inFromFile.nextLine());
lineScan.useDelimiter('/');
```
   b.
```
Scanner lineScan = new Scanner(inFromFile.nextLine());
lineScan.usedelimiter("/");
```
   c.
```
Scanner lineScan = new Scanner(inFromFile.nextLine(), "/");
```
   d. **None of the above**
4) What type of relationship exists between `Scanner` and `Iterator`?
   a. "uses" relationship with `Scanner` using the methods of `Iterator`
   b. "has a" relationship with `Scanner` containing references to `Iterator` objects
   c. **"implements" relationship with `Scanner` implementing the methods of `Iterator`**
   d. None of the above
5) Which of the following statements prints the second half of a line obtained from **in.txt** through the `inFromFile` object created above? Assume that the line contains an even number of characters.
   a.
```
String line = inFromFile.nextLine();
for(int i=line.length/2;i<line.length;i++)
        System.out.print(line.charAt(i));
```
   b.
```
int i; String line = inFromFile.nextLine();
for(int i=line.length()/2;i<line.length();i++)
        System.out.print(line.charAt(i));
```
   c. Both of the above
   d. **None of the above**
6) Assume that `line` is a variable that holds a line of text obtained from **in.txt**. Which of the following switch statements prints **too short** one time if `line` contains less than 2 characters (inclusive) and **too long** otherwise?
   a.
```
switch(line.length() <= 2) {
case true: System.out.print("too short");
case false: System.out.print("too long");}
```
   b.
```
switch(line.length()) {
case 0: System.out.print("too short");
case 1: System.out.print("too short");
case 2: System.out.print("too short");
default: System.out.print("too long");}
```
   c. Both of the above
   d. **None of the above**

7) When processing a file using the `Scanner` class, which of the following has to be placed at the end of the header of the `main` method?
   a. `throw IOException`
   b. **`throws IOException`**
   c. `Throw IOException`
   d. None of the above
8) When processing a file using the `Scanner` class, which of the following `import` declaration statements must be used?
   a.
   ```
   import java.util.*;
   import java.io.* ;
   ```
   b.
   ```
   import java.util.Scanner;
   import java.io.*;
   ```
   c. **Either of the above**
   d. None of the above
9) Which of the following provides `print` and `println` methods that can print data to a file?
   a. `FileWriter`
   b. `BufferedWriter`
   c. **`PrintWriter`**
   d. None of the above
10) Assume that `outToFile` is an object that can write data to the **out.txt** output file. Which of the following code fragments can be used to create an exact copy of the data contained in **in.txt** inside **out.txt**?
   a.
   ```
   while(inFromFile.hasNext())
          outToFile.println(inFromFile.nextLine());
   ```
   b.
   ```
   while(inFromFile.hasNext())
          outToFile.print(inFromFile.nextLine());
   outToFile.close();
   ```
   c. Both of the above
   d. **None of the above**
11) Not closing the object used to write data to a file once all the data is transferred to the output file is an example of a
   a. **Logical error**
   b. Syntax error
   c. Run-time error
   d. None of the above in the sense that not closing the object does lead to an error
12) Which of the following represents a proper way of creating a `FileWriter` object called `fw` that is attached to the `out.txt` output file?
   a. `FileWriter fw = new FileWriter(out.txt);`
   b. **`FileWriter fw = new FileWriter(new String("out.txt"));`**
   c. Both of the above
   d. None of the above
13) Which of the following is true about writing data to a file?
   a. Not closing the `PrintWriter` object results in an empty file
   b. The header of the main method must indicate that an `IOException` can be thrown if something goes wrong
   c. **Both of the above**
   d. None of the above
14) Which of the following sets of methods are part of `Iterator`?
   a. `hasNext(), nextLine()`
   b. `hasMoreTokens(), nextToken()`
   c. `hasNext(), next(), nextLine()`
   d. **None of the above**

Problem 2: True or false questions (**15 minutes)** [16 points]

1. The following code prints: `No ***`

```
boolean x = true;
if(!x)
      System.out.print("Yes");
else
      System.out.print("No");
      System.out.print(" ***");
```

Answer: **True** False

2. The output of the program segment below is the text: `not rich Done`

```
int rate = 6;
if(rate >= 9 || rate <=5)
if(rate <= 5) System.out.print("poor");
else System.out.print("middle class");
else System.out.print("not rich");
System.out.print("Done");
```

Answer: True **False**

3. The following method returns the leftmost digit of the input parameter `x`.

```
public int method1(int x) {
      while(x>10)
            x = x/10;
      return x;}
```

Answer: True **False**

4. The following code is syntactically valid in Java.

```
for(int z=10; z>=0; )
      z--;
System.out.print(z);
```

Answer: **True** False

5. To check if an integer variable `x` does not belong to the range of 3 (inclusive) through 5 (inclusive), we could use the following

```
if(!(x < 3) || !(x > 5))
```

Answer: True **False**

6. Consider the following code below using nested while loops:

```
int count1=1, count2=1;
while(count1<=100) {
      while(count2 <= 1000) {
            System.out.println(count2);
            count2++;}
      count1++;}
```

The code above is functionally equivalent to the following code using nested for loops:

```
for(int count1=1; count1<=100; count1++)
      for(int count2=1; count2<=1000; count2++)
            System.out.println(count2);
```

Answer: True **False**

7. The following method reverses the digits of the input parameter `x`.

```
public int method2(int x) {
      int reverse=0;
      do {
            reverse = reverse + x % 10;
      } while(x > 0);
      return reverse;}
```

Answer: True **False**

8. In Java, parameters are passed by value; therefore, changes made to primitive parameters are permanent and persist after the method is exited.

Answer: True **False**

9. The following code prints the odd numbers between 1 (inclusive) and 10 (inclusive).

```
int count=1;
while(count <= 10) {
      if(count % 2 == 0)
            continue;
      System.out.println(count);
      count++;}
```

Answer: True **False**

10. The following code prints the max value among the three variables `num1`, `num2`, and `num3`.

```
int val;
if(num1 > num2)
if(num1 > num3)
val = num1;
else
val = num3;
else if(num2 > num3)
val = num2;
else
val = num3;
System.out.print(val);
```

Answer: **True** False

11. The following code can be rewritten with a `switch` statement.

```
double val=2.5;
if(val == 3) System.out.print("First threshold reached");
else if(val == 2.5) System.out.println("2nd threshold reached");
else System.out.println("Neither thresholds reached");
```

Answer: True **False**

12. The following code prints: `8`

```
int x = 1;
for(int i=0; i<3; i++)
for(int j=i; j<3; j++)
x = x+j;
System.out.print(x);
```

Answer: True **False**

13. Consider a class called `ClassA` that contains one instance variable and one static variable. If three objects are created from this class, then there will be three different versions of the instance variable and three different versions of the static variable.

Answer: True **False**

14. The following code uses multiple overloaded versions of the `System.out.println` method.

```
int x = 1, y=2;
System.out.println(x);
System.out.println(y);
```

Answer: True **False**

15. The following code prints the gcd of `num1` and `num2`.

```
while(num1 != num2) {
      num1 = (num1 > num2) ? num1 – num2 : num2 – num1;}
System.out.print(num1);
```

Answer: True **False**

16. String is an example of a self-dependent class.

Answer: **True** False

Problem 3: Method analysis (**15 minutes**) [10 points]

Consider the methods given below.

1) What would be the output of the method call: `compute(12);`

```
public void compute(int n){
      int x = n;
      int c = 0;
      while(x>0)
        x = x%2;
        c++;
      for(int i=0; i<=c; i++)
        n = n + 2;

      System.out.print(n/2);
}
```

a. 6
b. 7
c. 8
d. Compile-time error
e. None of the above

2) What would be returned if it were called using the statement: `stringManip("aba");`?

```
   public static String stringManip(String S){

      String S1= "";
      String S2 = "";
      for(int i=0; i<S.length(); i++){
            for(int j=S.length(); j>i; j--){
                  if(S.charAt(i) == S.charAt(j-1)) {
                        S1=S1+S.charAt(i);}
                  else
                        S2=S2+ S.charAt(i) + S.charAt(j-1);
            }
      }
      S=S1 + S2;
      return S;
   }
```

a. aabbaa
b. aabaabba
c. abbaba
d. Compile-time error
e. none of the above

Problem 4: Code analysis (**15 minutes**) [10 points]

For each of the following code fragments, what is the value of **x** after the statements are executed?

(1)
```
int x = 0, v=12345, c=0;
while (v> 0){
  c = v % 10;
  if (c != 0 || c%2 !=0) x++;
  v = v / 10;}
```

Answer: x = 5

(2)
```
char c = 'E'; int x = 5;
switch(c) {
case 'B':
   x += 10;
break;
case 'D':
  x += 20;
case 'C':
  x += 30;
default:
  x += 40;
}
```

Answer: x= 45

(3)
```
int x = 0;
for(int i=0; i < 3; i = i+2)
for(int j=0; j<i; j++) x++;
```

Answer: x = 2

(4)
```
String S = new String("We rise by lifting others");
int y = 0; char c; String x = "";
do {
  c = S.charAt(y);
  if(c== 'r' || c == 'e’)
  x += c; y++;
} while(y < S.length());
```

Answer: x = “ereer”

(5) (2 pts)

```
int x=3, i=-1;
do {
x -= i;
i++;
} while(i < 5);
x /= i;
```

Answer: x = -1

(6) (2 pts)

```
String S = "1 2 3 4 5";
int x = 2, n;
Scanner scan = new Scanner(S);
while(scan.hasNext()) {
  n = Integer.parseInt(scan.next());
  if(n > x+1)
        x = n;
}
```

Answer x = 4

(7) (2 pts)

```
int v=30, x = 0;
for (int i=1; i<v; i++)
if(v%i == 0)  x+=i;
```

Answer x = 42

Problem 5: Coding (**50 minutes)** [50 points]

1. Write a program called `WordStats` that reads a sentence from the user and then prints to the screen the following statistics pertaining to that input sentence:
    - Number of upper-case letters, denoted by `NbU`
    - Number of lower-case letters, denoted by `NbL`
    - Number of digits, denoted by `NbD`
    - And number of white space characters, denoted by `NbS`

Sample output:

Enter a sentence: Exam is fun
Nb. of capital letters: 1
Nb. of lower case letters: 8
Nb. of digits: 0
Nb. of white space characters: 2

```
import java.util.Scanner;

class  WordStats{

        public static void main(String [] args)
        {

                String S;
                int NbU = 0, NbL = 0, NbD = 0, NbS = 0;

                Scanner scan = new Scanner(System.in);

                System.out.println("Enter a sentence: ");
                S = scan.nextLine();          // String does not contain special characters

                for(int i = 0; i < S.length(); i++)
                {
                   if(S.charAt(i) >= 'A' && S.charAt(i)<= 'Z') NbU++;
                        else if (S.charAt(i) >= 'a' && S.charAt(i)<= 'z') NbL++;
                        else if (S.charAt(i) >= '0' && S.charAt(i)<= '9') NbD++;
                        else NbS++;
                }

                System.out.println("Nb. of capital letters: " + NbU);
                System.out.println("Nb. of lower case letters: " + NbL);
                System.out.println("Nb. of digits: " + NbD);
                System.out.println("Nb. of white space characters: " + NbS);

        }
}
```

2. Write a program called `ComputePI` to compute the value of π, using the following series expansion.

$$\prod = 4 \times \left(1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \ldots \pm \frac{1}{N}\right)$$

The user has to enter the value of `N`. The program should then compare the computed value of π with the constant value `Math.PI` provided by the `Math` class and output the absolute value of the difference between the computed value π and `Math.PI` rounded to 4 decimal points.

Sample output:
Enter integer N: 6
Computed PI = 3.4667
The difference with actual PI: 0.3251

```
import java.util.Scanner;
import java.text.DecimalFormat;

class  ComputePI{
        public static void main(String [] args)
        {
                int N;
                double computedPI = 0;
                Scanner scan = new Scanner(System.in);

                do
                {       System.out.println("Enter integer N: ");
                        N = scan.nextInt();
                } N <=0;

                int sign = -1;

                for(int i = 1; i <= N; i+=2)
                {
                   sign *= -1;
                        computedPI += sign * (1/(double)i);
                }
                computedPI *= 4;
                DecimalFormat fmt = new DecimalFormat("0.####");
                double diff = Math.abs(Math.PI - computedPI);
                System.out.println("Computed PI = " + fmt.format(computedPI));
                System.out.println("The difference with actual PI = " + fmt.format(diff));
        }
}
```

3. Write a program called `NumberValidation` that reads from the user an `int` value. Your program should then determine whether or not the input value is valid by:
    a. Calculating the following two sums: `S1`, which is the sum of all the digits and `S2`, which is the sum of all the odd digits.
    b. And then verifying that `S1+S2` is divisible by 10.

    Note that the input number is valid if `S1+S2` is divisible by `10` and invalid otherwise.

Sample output:
Enter integer value: 56789
56789 is invalid

```
import java.util.Scanner;

class  NumberValidation{

        public static void main(String [] args)
        {

                int N, n, d, S1=0, S2=0;

                Scanner scan = new Scanner(System.in);

                System.out.println("Enter integer value: ");
                N = scan.nextInt();

                n = N;

                while(n != 0)
                {
                   d = n%10;
                        S1 += d;
                        if(d%2 !=0) S2 += d;
                        n = n/10;
                }

                if((S1+S2)%10 ==0) System.out.println(N + " is valid");
                else System.out.println(N + " is invalid");

        }
}
```

4. Write a program called `ToggleCharacters` that reads a word from the user and then prints out a modified version of that word, where every lower case letter of the original word is converted into its upper-case equivalent and every upper case letter of the original word is converted into its lower-case equivalent (See the sample output given below).

Sample output:
Enter a word: CreDIt
Modified word: cREdiT

```
import java.util.Scanner;

class  ToggleCharacters{

        public static void main(String [] args)
        {

                String word;


                Scanner scan = new Scanner(System.in);

                System.out.println("Enter a word: ");
                word = scan.nextLine();

                String word2 = "";

                for(int i=0; i< word.length(); i++)
                {
                        String S = "" + word.charAt(i);

                        if(word.charAt(i) >= 'A' && word.charAt(i) <= 'Z') word2 += S.toLowerCase();
                        else word2 += S.toUpperCase();

                }

                System.out.println("Modified word: " + word2);

        }
}
```

5. Write a program called `DuplicateElimination` that reads a word from the user denoted by `word`. Your program should then create a modified version of the input word denoted by `modifiedWord`, where all the letters occurring multiple times in the original word appear once in `modifiedWord`.

Sample output:
Enter a word: engineering
Modified word: engir

```
import java.util.Scanner;

class  DuplicateElimination{

        public static void main(String [] args)
        {

                String word, modifiedWord ="";
                String dup = "";                // special string to store duplicate characters

                Scanner scan = new Scanner(System.in);

                System.out.println("Enter a word: ");
                word = scan.nextLine();

                for(int i=0; i< word.length(); i++)
                {
                        if(modifiedWord.indexOf(word.charAt(i)) == -1)
                                modifiedWord += word.charAt(i);

                }

                System.out.println("Modified word: " + modifiedWord);
        }
}
```