ANSWERS Spring 2019 CMP 167 Final Exam V2

114 points total

Name:_____ Date:_____

Q1. (8 Points) What is the output of the following method calls? If there is an error say error

Use the code below to show the output for question	Method Call	Output

`public static void magic(int x, int y, int z){`
 	
`  if(x % 2 == 0) {`
`  double a = x / y;`
`  System.out.println("earth = "+ a);`
`  } else {`
`  double a = x / (double)z;`
`  System.out.println("mars = "+ a);`
`  }`
`}`
	magic(10, 0, 5);	ERROR or Infinity

	magic(9, 5, 0);	ERROR or Infinity
	magic(10, 3, 2);	earth = 3.0
	magic(9, 4, 2);	mars = 4.5

Q2. (8 Points) What is the output of the following method calls? If there is an error say error

Use the code below to show the output	Method Call	Output

`public static void funPrint(int a) {`
`  if( a % 3 != 0 ) {`
`  System.out.println("a = " + a);`
`  for(int i = 0 ; i < a ; i++){`
`  System.out.println("sky = " + i);`
`  }`
`  } else {`
`  for(int i = a ; i > 0 ; i=i-2) {`
`  System.out.println("sea " + i);`
`  }`
`  System.out.println("a = " + a);`
`  }`
`}`
	funPrint(6);	sea 6
		sea 4
		sea 2
		a = 6

	funPrint(4);	a = 4 sky = 0 sky = 1 sky = 2 sky = 3
	funPrint(3);	sea 3 sea 1 a = 3
	funPrint(0);	a = 0

Q3. (10 Points) What is the output of the following program?

CODE	OUTPUT

public class Switch {
  public static void main(String[] args) {
  int[] arr = { 8, 6, 5, 7, 9, 0, 4, 2, 1, 3 };
 	
  for (int i = 0; i < arr.length; i++) {
  int j = arr.length - i - 1;
  switch (i) {
  case 0:
  System.out.println("i = " + i + " : " + (arr[j - 1] * arr[i + 2]));break;
  case 1:
  System.out.println("i = " + i + " : " + (arr[j - 3] + arr[i + 4]));
  case 2:
  System.out.println("i = " + i + " : " + (arr[j - 4] / arr[i + 6])); break;
  case 3:
  System.out.println("i = " + i + " : " + (arr[j - 1] - arr[i + 3]));
  case 4:
  System.out.println("i = " + i + " : " + (arr[j - 2] * arr[i - 2]));break;
  case 5:
  System.out.println("i = " + i + " : " + (arr[j + 3] + arr[i - 3]));
  case 6:
  System.out.println("i = " + i + " : " + (arr[j + 1] / arr[i - 2]));break;
  case 7:
  System.out.println("i = " + i + " : " + (arr[j + 5] - arr[i - 1]));
  default:
  System.out.println("i = " + i + " : " + (arr[j + 7] * arr[i - 4]));
  }
  }
  }
}
	i = 0 : 5
	i = 1 : 0
	i = 1 : 4
	i = 2 : 7
	i = 3 : -4
	i = 3 : 54
	i = 4 : 35
	i = 5 : 7
	i = 5 : 0
	i = 6 : 1
	i = 7 : -2
	i = 7 : 21
	i = 8 : 9
	i = 9 : 0

Q4. (20 Points) **The Java class below has 10 errors.** It is supposed to read grades from the user until a negative number is entered, incrementing count, accumulating the sum, tracking min and max grade. After all grades have been entered, it should print the lowest grade, highest grade and average of all the grades.

In the second column mark each line of code either as "OK" , or write the correct syntax for that line of code.

`import java.util.Scanner;`	ok
`public class ErrorProne {`	ok
`public static void main(String args) {`	public static void main(String [ ] args) {
`Scanner scnr = new Scanner(Standard);`	Scanner scnr = new Scanner(System.in);
`int gradeMin = -1;`	int gradeMin = Integer.MAX_VALUE;
`int gradeMax = -1;`	int gradeMax = Integer.MIN_VALUE;
`int gradeSum = 0;`	ok
`int gradeCount = -0;`	int gradeCount = 0;
`int grade;`	ok
`while ((int grade = scnr.nextInt()) >= 0) {`	while ((grade = scnr.nextInt()) >= 0) {
`if (grade <= gradeMax) {`	if (grade >= gradeMax) {
`gradeMax = grade;`	ok
`}`	ok
`if (grade >= gradeMin) { gradeMin = grade; }`	if (grade <= gradeMin) {
`gradeSum += grade;`	ok
`gradeCount--;`	gradeCount++;
`}`	ok
`double gradeAverage = (double) gradeSum/gradeCount;`	ok
`System.out.println("Minimum Grade = " , gradeMin);`	System.out.println("Minimum Grade = " + gradeMin);
`System.out.println("Maximum Grade = " + gradeMax);`	ok
`System.out.println("Average Grade = " + gradeAverage);`	ok
`scnr.close();`	ok
`}`	ok
`}`	ok

Q5. (10 Points) Write a method that asks the user for a sentence using the Scanner.
Use the String class's **split** method to split the text into an array of Strings using " " as the delimeter.
Return a String containing the words of the sentence in reverse order with all characters converted to uppercase.

Example Run:

Enter a sentence:
User enters: "Today is the Final Exam"
String returned will be: "EXAM FINAL THE IS TODAY"

```
public static String question5_V2(){
        Scanner scnr = new Scanner(System.in);
        System.out.println("Enter a sentence:");
        String sentence = scnr.nextLine();
        String [] tokens = sentence.split(" ");

        String rev = "";
        for(int i=tokens.length-1; i>0; i--){
                rev += (tokens[i].toUpperCase()+" ");
        }
         rev += tokens[0].toUpperCase();
         scnr.close();
         return rev;
    }
```

Q6. (56 Points) Complete a **public class** to represent a **Building** as described below.
(a-e 4 pts each) (f-h 8 pts each) (i, j 5 pts each) (k 2 pts)

a. Create the following private member variables

int numWindows	String ownerName	boolean hasStairs	String [ ] roomNames

```
private int numWindows;
private String ownerName;
private boolean hasStairs;
private String [] roomNames;
```

b. Create the default constructor so it will initialize the variables to the following default values.

Attribute	numWindows	ownerName	hasStairs	roomNames
Default Value	0	“Doe”	false	initialize to length of 5

```
public Building(){
        numWindows = 0;
        ownerName = "Doe";
        hasStairs = false;
        roomNames = new String[5];
    }
```

c. Create the fully overloaded constructor that accepts all variables as input parameters.

```
    public Building(int numWindows, String ownerName, boolean hasStairs,
                    String[] roomNames) {
        this.numWindows = numWindows;
        this.ownerName = ownerName;
        this.hasStairs = hasStairs;
        this.roomNames = roomNames;
    }
```

d. Create **setter** method only for the **ownerName**.

```
public void setOwnerName(String name){
        ownerName = name;
    }
```

e. Create **getter** methods for all the variables.

```
public int getNumWindows() {
        return numWindows;
}


public String getOwnerName() {
        return ownerName;
}


public boolean getHasStairs() {
        return hasStairs;
}


public String[] getRoomNames() {
        return roomNames;
}
```

f. Create a method that allows the name of a room at an index in the roomNames array to be changed. (Note: Only update the roomNames array if the index is valid)

```
public void changeRoomName(int index, String updatedRoomName){//f
        if(index >=0 && index<roomNames.length){
                roomNames[index] = updatedRoomName;
        }
}
```

g. Create a helper method that determines the equality of two String arrays and returns a boolean, by comparing the value at each index location. Return true if all elements of the arrays matches, return false if there is any mismatch. (Note: this method will be used by the equals method in part h)

```
private boolean doArraysMatch(String[] arr1, String[] arr2){//g
		if(arr1.length == arr2.length){
			for(int i=0; i<arr1.length; i++){
				if( ! arr1[i].equals(arr2[i]) ){
					return false;
				}
			}
			return true;
		}
		return false;
}
```

h. Create a helper method that gets the array of roomNames as a comma separated String.

```
private String getRoomNamesAsString(){
		String s = "";
		for(int i=0; i<roomNames.length-1; i++){
			s += roomNames[i] + ", ";
		}
		s += roomNames[roomNames.length -1];
		return s;
}
```

i. Create the **toString()** method for the Building object such that it returns a well structured sentence containing the information for all the object's variables.
(Note: use the helper method getRoomNamesAsString from part i**)**

Example of the String returned:
"Building: Number of windows = 8, Owner = Hulk, has stairs, Rooms: living room, dining room, bedroomA, bedroomB, kitchen"

```
@Override
	public String toString() {//i
		return "Building: numWindows= " + numWindows +
				", ownerName= " + ownerName +
				", hasStairs= " + hasStairs +
			", roomNames= " + getRoomNamesAsString();
	}
```

j. Create the **equals(Object o)** method for the Building object, such that it returns true if the values of all the members of both the calling object and the passed in object match. Return false if any values do not match. (Note: Use the helper method named doArraysMatch from part h)

```
@Override
	public boolean equals(Object o){//j
		if(o instanceof Building){
			Building otherBuilding = (Building)o;
			if(this.numWindows == otherBuilding.numWindows){
				if(this.ownerName.equals(otherBuilding.ownerName)){
					if(this.hasStairs == otherBuilding.hasStairs){
						if(doArraysMatch(this.roomNames, otherBuilding.roomNames)){
							return true;
						}
					}
				}
			}
		}
		return false;
	}
```

k. Create 2 instances of Building objects using your overloaded constructors from 6c with the following values.

Example of the values for b1, and b2 below:

b1	
numWindows	8
ownerName	Hulk
hasStairs	true
roomNames	living room, dining room, bedroomA, bedroomB, kitchen

b2	
numWindows	6
ownerName	Batman
hasStairs	false
roomNames	living room, dining room, kitchen, cave1, cave2

```
Building b1 = new Building(8, "Hulk", true, new String[ ]{"living room", "dining room", "bedroomA",
"bedroomB", "kitchen"} );
Building b2 = new Building(6, "Batman", false, new String[ ]{"living room", "dining room","kitchen", "cave1",
"cave2"} );
```