



# University of Maryland College Park

## Department of Computer Science

### CMSC131 Spring 2022

### Exam #2

FIRSTNAME, LASTNAME (PRINT IN UPPERCASE): **KEY**

STUDENT ID (e.g. 123456789):

#### Instructions

- Please print your answers and use a pencil.
- Do not remove the staple from the exam. Removing it will interfere with the Gradescope scanning process.
- To make sure Gradescope can recognize your exam, print your name, write your directory id at the bottom of pages with the text DirectoryId, provide answers in the rectangular areas provided, and do not remove any exam pages. Even if you use the provided extra pages for scratch work, they must be returned with the rest of the exam.
- This exam is a closed-book, closed-notes exam, with a duration of 50 minutes and 100 total points.
- Your code must be efficient.
- Multiple choice questions have only one answer unless indicated otherwise.
- You don't need to use meaningful variable names; however, we expect good indentation.

#### Grader Use Only

#1	Problem #1 (Short Answers - 2pts each)	16
#2	Write a method	14
#3	Write a class	70
<b>Total</b>	Total	100

## Problem #1 (Short Answers – 2 pts each)

- (2 pts) A **class** is a blueprint to make objects.
- (2 pts) **pseudocode** is English-like description of the set of steps required to solve a problem.
- (2 pts) The term **abstraction** is used in CS to describe what a method/data structure does, not how (i.e. don't worry about the implementation details).
- (2 pts) Which statement is true?
  - All fields in a class have to be static.
  - A static method in a class can access instance variables of the current object.
  - The default value for a reference field is null.**
  - A class will not compile unless the programmer explicitly writes a constructor.
- (2 pts) Which statement is false?
  - A class can have more than one constructor
  - The keyword `this` can be used to access the current object in an instance method
  - The default behavior of the equals method is to behave like the `==` operator.
  - The `break` keyword has only one use in Java; to break out of a loop.**
- (2 pts) Assume the following code fragment in a `main` method, what is the output?

```
int sum = 0;
for (int i=0; i<7; i++) {
    if (i % 3 ==0)
        continue;
    sum+=i;
}
System.out.println(sum);
```

12

- (2 pts) If there is code that needs to run regardless of whether the `try` clause completes or a `catch` clause handles an exception, you can put in a **finally** block.
- (2 pts) Assume the following code fragment in a `main` method. If it will not compile, write Error. Otherwise, write the output.

```
int x =5;
int y = x!=5? 7: 8;
System.out.println(y);
```

8

## Problem #2 (Write a method – 14 pts)

- Complete the code for the method below. This method takes in an array of `Strings` (you can assume there will be at least one `String` and no element is assigned `null`). It returns the longest `String` (i.e. most number of characters) in the array. If there are more than one `String` in the array with the same longest length, it returns the first one encountered when visiting elements starting from the first element (i.e. index 0). Look at the sample call below, for how it should work:

```
String ex [] = {"java", "if", "while", "for"};
System.out.println(longestStr(ex)); //will print while
```

```
public static String longestStr(String s [])
{
    int len =0;
    int index=0;

    for (int i =0; i<s.length; i++)
    {
        if (s[i].length()> len) {
            len=s[i].length();
            index = i;
        }
    }
    return s[index];
}
```

### Problem #3 (Write a class)

Complete the implementation of a class called `CSExam` that represents an exam in a CS class. A `CSExam` has a name (`name` instance variable), an array that holds the points associated with each of the three parts of a `CSExam` (`subParts` instance variable), and a total of all the points on the exam (`total` instance variable). The number of `CSExam` objects created is represented by the static field `TOTAL_EXAMS`. Below there is a driver that illustrates some of the functionality associated with the class. Feel free to ignore it if you know what to implement. You **MAY NOT** add any methods beyond the ones specified below (not even private). Also you **MAY NOT** add any additional instance or static variables. Local variables in your methods are fine. All the methods below are public and non-static unless specified otherwise.

```
public class CSExam {
    private String name;
    private int [] subParts;
    private int total;
    private static int TOTAL_EXAMS = 0;
}
```

Driver	Driver Output
<pre>public class Driver {      public static void main(String[] args) {         CSExam e1= new CSExam("132Exam", 25, 25, 25);         CSExam e2 = new CSExam();         CSExam e3 = new CSExam(e1);          e3.setParts(1, 51);         e3.setParts(1, -51); //invalid - no change          int[] num = e3.getSubParts();         num[0] =num[1] =100; //change only to local copy          System.out.println(e1);         System.out.println(e2);         System.out.println(e3);          System.out.println(CSExam.getTotal_EXAMS());          System.out.println(CSExam.isUnbalanced(e1));         System.out.println(CSExam.isUnbalanced(e3));          System.out.println(e1.equals(e3));         e1.setParts(1, 51);         System.out.println(e1.equals(e3));      } }</pre>	<pre>132Exam, 25, 25, 25, 75 CMSC131Exam2, 0, 0, 0, 0 132Exam, 25, 51, 25, 101 3 false true false true</pre>

1. Define a **constructor** that takes a string parameter called `name` and three integer parameters called `part1`, `part2`, and `part3`. The parameters are considered invalid if the `name` parameter is null or if any of the 3 integers parameter are negative (you can assume an integer is passed in). If the parameters are invalid, throw the `IllegalArgumentException` with the message `Bad input data`. Assuming valid parameters, assign the `name` parameter to the `name` field and **create a 3 element integer array** to be assigned to the `subParts` field where the first element will be assigned the value in `part1`, the second element will be assigned the value in `part2`, and the third element

will be assigned the value in `part3`. The `total` field will be assigned the sum of the elements of the array. Make sure you adjust the total number of objects created so far.

```
public CSExam(String name, int part1, int part2, int part3) {  
    if(name == null || part1 < 0 || part2 < 0 || part3 < 0) {  
        throw new IllegalArgumentException("Bad input data");  
    }  
    this.name = name;  
  
    subParts = new int[3]; //all are zero  
    subParts[0]= part1;  
    subParts[1]= part2;  
    subParts[2]= part3;  
  
    total =0;  
    for (int i =0; i <subParts.length; i++)  
    {  
        total +=subParts[i];  
    }  
  
    TOTAL_EXAMS++; //new Exam object  
  
}
```

2. Define a **default constructor** that initializes a `CSExam` with the name "CMSC131Exam2" and the 3 parts to 0. You must call the previous constructor in order to implement this constructor, otherwise you will not get any credit.

```
public CSExam() {  
    this("CMSC131Exam2", 0, 0, 0);  
}
```

Directory id:

3. Define a **copy constructor** method for the class. Changes to the copy should not affect the original object.

```
public CSExam(CSExam myExam) {
    this.name = myExam.name;

    this.subParts = new int[3]; //all are zero
    for (int i =0; i <subParts.length; i++)
    {
        this.subParts[i]=myExam.subParts[i];
    }
    this.total= myExam.total;

    TOTAL_EXAMS++; //new Exam object
}
```

4. Define a method called `setParts` that has two parameters: a integer called `index` and an integer called `newVal`. The method will change the value stored at `index` of the `subParts` array to `newVal` assuming that the parameters are valid. The parameters are valid if the `index` parameter is within the bounds of the array and the `newVal` parameters is not negative. Don't forget to update `total`. If the parameters are invalid, no change should take place. The method returns a reference to the current object.

```
public CSExam setParts(int index, int newVal)
{
    if (index>=0 && index<=2 && newVal>=0)
    {
        subParts[index] =newVal;
        total =0;
        for (int i =0; i <subParts.length; i++)
        {
            total +=subParts[i];
        }
    }
    return this;
}
```

5. Define a **static** method called `getTOTAL_EXAMS` that simply returns the value of the static field `TOTAL_EXAMS`.

```
public static int getTOTAL_EXAMS() {  
    return TOTAL_EXAMS;  
}
```

6. Define a method called `getSubParts` with no parameters that returns a copy of the `subParts` avoiding a privacy leak.

```
public int[] getSubParts()  
{  
    int [] copy = new int [3];  
    for (int i =0; i <subParts.length; i++)  
    {  
        copy[i]=this.subParts[i];  
    }  
    return copy;  
}
```

7. Define a `toString()` method that returns a string with the name, the elements in the `subParts` array, and the `total` each separated by a comma (See output of the driver). Any number of spaces (including 0) between the values is OK.

```
public String toString() {  
    return name + ", " + subParts[0] + ", " + subParts[1] + ", " +  
    subParts[2] + ", " + total;  
}
```

Directory id:

8. Define an `equals` method for the class. The parameter must be of type `Object` and you should use the `getClass` method not the `instanceof` operator in your code. Two `CSExams` are considered equal if they have the same name, same values in the corresponding elements of the `subParts` array, and the same `total`.

```
public boolean equals(Object obj) {
    if (this == obj)
        return true;
    if (obj == null || getClass() != obj.getClass())
        return false;
    CSExam other = (CSExam) obj;

    return name.equals(other.name) &&
        subParts[0] == other.subParts[0] &&
        subParts[1] == other.subParts[1] &&
        subParts[2] == other.subParts[2] &&
        total == other.total;
}
```

9. Define a **static** method called `isUnbalanced` that returns a `boolean` and has one parameter: a `CSEexam` called `e`. It will return `true` if the sum of **any 2 elements** of the `subParts` array of the object passed in are strictly less than the third. If that is not the case, return `false`.

```
public static boolean isUnbalanced(CSEexam e)
{
    if (e.subParts[0]+e.subParts[1] < e.subParts[2] ||
        e.subParts[1]+e.subParts[2] < e.subParts[0] ||
        e.subParts[0]+e.subParts[2] < e.subParts[1] )
        return true;
    return false;
}
```

**EXTRA PAGE IN CASE YOU NEED IT**

**LAST PAGE**