Midterm Exam 2 Wednesday Answer Sheet
CSC 210.03, WED 11/18/2019, 12:35 – 1:45 PM
Midterm Exam 2: 100 points
Full Name: Kullathon Sitthisarnwattanachai
SFSU ID: 921425216

Show your work, typed up or using pens. Your answers must be clear and readable to the grader(s).

HONOR CODE: - Please follow CS Department’s policies: https://cs.sfsu.edu/cheating-and-plagiarism-policy

No.	Answers	Points

1a	In the Employee class:
	`Employee(String employeeName) {`
	`    this.employeeName = employeeName;`
	`}`
		4

1b	Also in the constructor from part (1)(b):
	`Employee(String employeeName) {`
	`    this.employeeName = employeeName;`
	`    employeeCount++;`
	`}`
	Note that this must be the only constructor defined for the Employee class for this to work properly. Alternately, the incremental statement must be present throughout all constructors in the class.
		4

1c	In the Employee class:
	`public String getEmployeeName() {`
	`    return employeeName;`
	`}`
		4

1d	In the main method of the Test class (the main class): `System.out.println(x.getEmployeeName());`	4

1e	In the Employee class:
	`public void setEmployeeName(String employeeName) {`
	`    this.employeeName = employeeName;`
	`}`
		4

2a	Note that the comparison is case-sensitive. The method returns a blank string (“”) if the provided argument does not exactly match the expected keyword.
	`public class Question2a {`
	`    public static void main(String[] args) {`
		3

```java
        String returnValue = returnGreeting("Coding");
        System.out.printf("Return value: %s%n", returnValue);
    }

    static String returnGreeting(String kw) {
        return kw.equals("Coding") ? "Happy Coding!" : "";
    }
}
```

2b

Note that the comparison is case-sensitive. The method will not output anything if the provided argument does not exactly match the expected keyword.

```java
public class Question2b {
    public static void main(String[] args) {
        makeGreeting("Coding");
    }

    static void makeGreeting(String kw) {
        if (kw.equals("Coding")) {
            System.out.println("Happy Coding!");
        }
    }
}
```

3

2c

```java
public class Question2c {
    public static void main(String[] args) {
        displayGreeting();
    }

    static void displayGreeting() {
        System.out.println("Happy Coding!");
    }
}
```

3

2d

```java
import java.util.Scanner;

public class GreetingHub {
    public static void main(String[] args) {
        if (promptEvent().equals("Coding")) {
            System.out.println("Happy Coding!");
        }
    }

    static String promptEvent() {
        Scanner scan = new Scanner(System.in);
        System.out.print("Enter the event: ");
        return scan.nextLine();
    }
}
```

7

3a

The following are the field declaration for the SFSUClub class.

Note that the class declaration has been modified from that given by the question prompt. The “public” modifier has been removed from the declaration to ensure that the file can be complied. There exists another class “SFSUClubDriver” which the main class declared with the “public” modifier in the same class.

5

```java
class SFSUClub {
    private static int numOfClubs;
    private String name;
    private int numOfMembers;
}
```

The variable that is being used to keep track of the number of clubs is called “numOfClubs.” The naming of the variable is made to be consistent with other variables that were already defined.

The variable is made private to prevent tampering. For the value to be accessed outside of the class, a getter method would be defined for the variable to ensure explicit access.

The variable is static so that it is part of the class itself and not a field of a particular instance. This is so the value updates across upon instantiations of the class.

This variable would be incremented in each of the constructors of the SFSUClub to make sure that it is being counted properly upon each instantiation.

3b

In the SFSUClub class:

```java
SFSUClub() {
    numOfClubs++;
}

SFSUClub(String name, int numOfMembers) {
    this.name = name;
    this.numOfMembers = numOfMembers;
    numOfClubs++;
}
```

5

3c

```java
public static void main(String[] args) {
    SFSUClub club1 = new SFSUClub();
    SFSUClub club2 = new SFSUClub("Coding Club", 20);
}
```

The no-arg constructor would be used when the value of the fields is not yet known. For example, if the program is expecting the user to enter the name of club or to change them later, a no-arg construct would be used to instantiate the object, and then the setters for the respective fields would be used later on.

If all fields are known, then one may choose to use the other constructor.

5

3d

In the SFSUClub class:

```java
public String getName() {
    return name;
}

public void setName(String name) {
```

6

	`    this.name = name;`
	`}`
	
	`public int getNumOfMembers() {`
	`    return numOfMembers;`
	`}`
	
	`public void setNumOfMembers(int numOfMembers) {`
	`    this.numOfMembers = numOfMembers;`
	`}`	

3e	A data field should be private if the field is only used within the class or if you want to prevent access outside the scope of the class.	2
3f	A data field should be static if you require the field to be consistent throughout all instances of the class. For example, in the questions above, the number of clubs were being tracked as a static field for the field to be updated and be consistent across all instances of the SFSUClub. The field can also be accessed without instantiating the class since it is not concerned with a particular instance. Do note that for the example used for the SFSUClub, the variable would have to be accessed via a static setter because the field is made private. Otherwise, the field can be access directly using the dot notation on the SFSUClub class.	2
4	Abstraction and encapsulation ensure the separation of how the class is implemented from those outside the scope of the class. This principle ensures that the class is functional without needing to understand the exact specifics of how the "inner-working" operates. For example, we use the Scanner class all the time to read input from the user. We do not need to understand exactly how the Scanner class obtains the user input from the buffer (for this class, anyway), but we do know that we can use it to obtain user input from stdin.	2
5	No. CSC210Exam already has a constructor defined. JVM will not create a default no-arg constructor.	2

6a	`for (String[] x : animalArray) {`
	`    for (String animal : x) {`
	`        System.out.printf("%s ", animal);`
	`    }`
	`    System.out.println();`
	`}`
	
	The outer for-each loop goes through the array to access each element in the array, which is also an array. The inner loop then loops through said array, where each of the elements are a string of animal names. The inner loop prints the name of each animal, and upon exhaustion reaches the end of the outer loop and printing a newline.
		5

6b	The row is the number of arrays in the array. Therefore, you can access its "length" directly using the dot notation. Below is the code that prints the row (prints "4"). `System.out.println(animalArray.length);`	3

6c

To get the number of columns, one must find the greatest number of columns from each of the row. In this case, we see that the first row has the greatest number of columns. Therefore, it must be the number of columns for the entire animalArray. Below is the code that checks for the greatest number of column (“col”) and prints it out (prints “4”).

```java
int col = 0;
for (String[] x : animalArray) {
    col = Math.max(x.length, col);
}
System.out.println(col);
```

Note that Math is part of java.lang and does not need to be imported.

2

7a

A class can be said to be a template from which objects are created. A class defines the type of values and properties of an object and how they behave. Meanwhile, an object, referred to as *an instance* of a class, holds properties that are specific to its instance, but are bound by the “rules” of the class. Two distinct objects of the same class may have different properties and can behave differently depending on those properties.

5

7b

In the SFSUClub class:

```java
public String getName() {
    return name;
}

public void setName(String name) {
    this.name = name;
}

public int getNumOfMembers() {
    return numOfMembers;
}

public void setNumOfMembers(int numOfMembers) {
    this.numOfMembers = numOfMembers;
}
```

5

7c

In the main method of the SFSUClubDriver class (the main class):

```java
for (SFSUClub club : clubArray) {
    System.out.printf("%s %d%n", club.getName(),
club.getNumOfMembers());
}
```

5

7d

In the SFSUClubDriver class (the main class):

```java
public static double meanMemberSize(SFSUClub[] clubs) {
    double sum = 0;
    for (SFSUClub club : clubs) {
        sum += club.getNumOfMembers();
```

5

	```
	    }
	    return sum / clubs.length;
	}
	```
7e	In the main method of the SFSUClubDriver class (the main class): `System.out.printf("Average club size: %.2f%n", meanMemberSize(clubArray));` Note that this is a continuation of (7)(c), and the "clubArray" is defined to be the array provided for that question.	5