

# CMSC131

## Midterm #2

First Name: \_\_\_\_\_

Last Name: \_\_\_\_\_

University ID: \_\_\_\_\_

**I pledge on my honor that I have not given or received any unauthorized assistance on this examination.**

Your signature:

\_\_\_\_\_

### General Rules:

- This exam is closed-book and closed-notes.
- Total point value is 100 points.
- For the problems where you need to write code, you do not need to worry about commenting, about runtime efficiency or about using descriptive variable names.
- **WRITE NEATLY.** If we cannot understand your answer, we will not grade it (i.e., 0 credit).

### Grader Use Only:

Page 2		(14)
Page 3		(14)
Page 4		(14)
Page 5		(9)
Page 6		(8)
Page 7		(10)
Page 8		(8)
Page 9		(8)
Page 10		(4)
Page 11		(11)
		(100)

1. (2 points) What keyword in Java refers to the current object in an instance method?
2. (2 points) Consider a class called **Car** that has a private instance variable called **Engine**. You are writing a “getter” for the **Engine**. Under what circumstances should you return a reference to a *copy* of the **Engine** rather than a reference to the existing **Engine**?
3.
  - A. (2 points) Suppose you have written a class called **Bird** and an interface called **CanSpeak**. Complete the class declaration below so that the class **Parrot** inherits all of the features from the **Bird** class and also supports the **CanSpeak** interface. (Fill in the blank).

`public class Parrot _____ {`

- B. (2 points) Assume that the **Parrot** class inherits a method called **chirp**. You decide that Parrots should chirp differently from ordinary birds, so you decide to replace the inherited **chirp** method with one of your own design (using the exact same method prototype). What do we call this?
  - C. (2 points) Which of the following code fragments make sense? (Circle “OK” or “No Way” for each statement.)

<code>Bird a = new Parrot();</code>	OK	NO WAY
<code>Parrot b = new Bird();</code>	OK	NO WAY
<code>CanSpeak c = new Parrot();</code>	OK	NO WAY
<code>Parrot d = new CanSpeak();</code>	OK	NO WAY

4. (2 points) If you need to store a sequence of characters that you will be manipulating repeatedly, what class should you use instead of the **String** class?
5. (2 points) Java arrays are always mutable. TRUE/FALSE (Circle one).

6. (4 points) Assume there is a class called `Bus` in a package called `vehicles`.
- What should the first statement at the top of the `Bus` class look like?
  - What statement could you put at the top of a source-code file so that you could just write “`Bus`” throughout, instead of repeatedly typing out the fully qualified name?
7. (3 points) Re-write the following code fragment as a single statement that contains only *one* assignment operator. (Hint: Use the ternary `?:` operator.)
- ```
if (x < 4) {  
    y = x + 2;  
} else {  
    y = 9;  
}
```
8. (3 points) Decide which pairs of methods are examples of proper “overloading”. For each pair, write either “**OK**” or “**NO WAY**”.
- `public void f(String a, Cat b)`  
`private String f(String x, Cat y)`
  - `public String g(int x)`  
`public String g(long y)`
  - `private int h(int a, String b)`  
`public int h(String x, int y)`
9. (2 points) Where can you use the keyword “`this`”? (Circle all that apply.)
- |                       |                     |                         |                   |
|-----------------------|---------------------|-------------------------|-------------------|
| <b>static methods</b> | <b>constructors</b> | <b>instance methods</b> | <b>interfaces</b> |
|-----------------------|---------------------|-------------------------|-------------------|
10. (2 points) What is the name of the popular interface that contains the `compareTo` method?

11. (2 points) What feature of the Java wrapper classes is illustrated by this code fragment:

```
Double[] a = {5.7, 6.2, 17.48};
```

12. (2 points) What do we call a variable that can reference different classes of objects at different times?

13. (2 points) What is the name of the Java class that serves as a “wrapper” for the primitive type `int`?

14. (2 points) How many *objects* are instantiated (created) by the following code fragment? Count carefully!

```
String[] a = new String[7];  
String[] b = a;
```

15. (4 points) What is the output from the following code fragment?

```
int j = 50;  
do {  
    j -= 2;  
    if (j > 20) {  
        continue;  
    }  
    if (j == 16) {  
        break;  
    }  
    System.out.println(j);  
} while (j >= 0);
```

16. (2 points) What is the name of the package that is automatically imported (in its entirety) into every Java class we write?

17. (9 points) Determine the output for the following program:

```
private static void f(String a) {
    try {
        int x = 15 / a.length();
        System.out.println("G");
    } catch (ArithmeticException e) {
        System.out.println("F");
    } finally {
        System.out.println("E");
    }
}

public static void main(String[] args) {
    try {
        f("Hello");
        f("");
        f(null);
        System.out.println("D");
    } catch (NullPointerException e) {
        System.out.println("C");
    } catch (ArithmeticException e) {
        System.out.println("B");
    }
    System.out.println("A");
}
```

**Write your answer here:**

18. (8 points) Consider the following code fragment. Assume that the variables `a` and `p` are type `int`.

```
if (a >= 3 && a <= 5) {  
    p = 777;  
} else if (a == 10) {  
    p = 888;  
} else if (a == 50 || a == 60){  
    p = 999;  
} else {  
    p = 0;  
}
```

Fill in the *switch* statement below so that it is equivalent to the code fragment above. You may not write any statements before or after the switch statement. Your switch statement should not have any unnecessary redundancy. You may not have any if/else statements or uses of the ternary `?:` operator in your answer.

```
switch(a) {
```

```
}
```

19. Assume that there is a **mutable** class called *Fish*. Consider the class below, which is only partially shown.

```
public class Aquarium {  
    private Fish[] fishArray;  
  
    public Fish[] getterOne() {  
        Fish[] copy = new Fish[fishArray.length];  
        for (int i = 0; i < fishArray.length; i++) {  
            copy[i] = new Fish(fishArray[i]);  
        }  
        return copy;  
    }  
  
    public Fish[] getterTwo() {  
        Fish[] copy = new Fish[fishArray.length];  
        for (int i = 0; i < fishArray.length; i++) {  
            copy[i] = fishArray[i];  
        }  
        return copy;  
    }  
  
    public Fish[] getterThree() {  
        return fishArray;  
    }  
}
```

- a. (2 points) What kind of copy is made in `getterOne`?

**REFERENCE / SHALLOW / DEEP** (Circle one.)

- b. (2 points) What kind of copy is made in `getterTwo`?

**REFERENCE / SHALLOW / DEEP** (Circle one.)

- c. (2 points) What kind of copy is made in `getterThree`?

**REFERENCE / SHALLOW / DEEP** (Circle one.)

- d. (2 points) Which of these getters would allow someone to *modify* a particular *Fish* that is part of an existing *Aquarium*?

**getterOne      getterTwo      getterThree** (Circle all that apply.)

- e. (2 points) Which of these getters would allow someone to *replace* a particular *Fish* that is part of an existing *Aquarium* with a *Fish* of their choosing?

**getterOne      getterTwo      getterThree** (Circle all that apply.)

- f. (8 points) Write an instance method for the Aquarium class called `removeFirstFish` with the prototype you see below. The method will remove the first fish in the aquarium (the first entry in the array), so that the Aquarium has one fewer fish. (The resulting array must be smaller. You may not use `ArrayList` or any of the other Java collections.)

If the aquarium is empty when this method is called then it should throw an `IllegalStateException`.

```
public void removeFirstFish() {
```

```
}
```



20. Consider this class:

```
public class IceCreamCone {  
  
    private int numberOfScoops;  
    private String flavor;  
  
    public IceCreamCone(int scoops, String flavor) {  
        this.numberOfScoops = scoops;  
        this.flavor = flavor;  
    }  
}
```

- a) (2 points) Write another constructor for this class that takes one parameter (the flavor), and sets the number of scoops to 2 every time. **Your constructor must make a call to the constructor shown above!**
- b) (6 points) Write an equals method for the IceCreamCone class. Two ice cream cones will be considered equal if they contain the same number of scoops and also the same flavor. Be sure that your equals method is an override not an overload!

21. (2 points) In which of the following cases will a “finally” block run? (Circle all that apply)

- a. When an exception is thrown *before* the try block that corresponds to the finally block.
- b. When an exception is thrown during the try block, and is caught locally
- c. When an exception is thrown during the try block, and it is not caught locally
- d. When a return statement is encountered during the try block
- e. When an exception is thrown during one of the catch blocks

22. (2 points) When using the Eclipse debugger: If you want to execute the current statement (the one that is green) and then stop at the very next statement, which button do you press? (Circle One).

**Step Into**

**Step Over**

**Step Return**

**Step Sister**

**DubStep**

23. (11 points) Fill in the method below. The return value will be a “ragged” 2-dimensional array. The lengths of the rows and their contents are determined by the parameter, as described below:

For example, if the parameter is the array  $\langle 3, 7, 5 \rangle$  then the return value must be the ragged 2-dimensional array depicted below:

```
3 3 3           // three copies of 3
7 7 7 7 7 7 7   // seven copies of 7
5 5 5 5 5       // five copies of 5
```

```
public static int[][] expander(int[] values) {
```