Multiple Choice Questions:

1. Topic: overflow/underflow

We know that a byte range is from -2^7 to (2^7) - 1 inclusive. Find the final amount of teeth.

```
public static void main(String[] args) {

   byte teeth = 127, tooth = 3;

   teeth += tooth;
   teeth -= tooth*2.5;

   System.out.println("Number of teeth is " + teeth);
}
```

a) 122
b) 123
c) 124
d) runtime error
e) 0

Solution:

We know that if we go above the limit of a primitive data type we cause an overflow and below the limit an underflow.

The value just wraps around.
Ex: for a byte the range is -128 to 127, so if we add 1 to 127 we get -128, and subtract 1 from -128 we get 127, and so on.

To solve this question we must first know what -2^7 to (2^7) - 1 is, which is -128 to 127. Knowing this we know when a value would overflow/underflow.

Doing the math with the teeth and tooth variables:

127 + 3 = -128 + 2 = -126 = teeth // first part done

-126 - (3*2.5) = -126 - 7.5 = 126 - 7
(we lose the fractional portion due to it being of type byte)
126 - 7 = -128 - 5 = 127 - 4 = 123

Therefore final result is: b) 123

2. Topic: Classes, specifically attributes (variables)

Review Question: Which of the following type of variables is accessible by another class in the same file?

A. Private
B. Protected
C. Default
D. Public

Solution:

Only private is class-only access and therefore is not accessible by another class even if it is in the same file.

3. Topic: Arrays

What is the output of this code?

```
int[] myArray  = {1, 2, 3, 4, 5, 6, 7, 8, 9};
int[] newArray = new int[9];
for (int i = 0; i < myArray.length - 1; i++){
      newArray[i] = myArray[i];
}
for (int i = newArray.length-1; i >= 0; i--){
      System.out.print(newArray[i] + “, ”);
}
```

A. 9, 8, 7, 6, 5, 4, 3, 2, 1,
B. 8, 7, 6, 5, 4, 3, 2, 1
C. 0, 8, 7, 6, 5, 4, 3, 2,
D. 0, 8, 7, 6, 5, 4, 3, 2, 1,

Solution: D
This problem should be easy if you understand how to properly iterate through an array. Both arrays are initialized to size 9, but when the contents of myArray are assigned to newArray, the last item is not moved over, because the bounds of the for loop are set to (myArray.length-1) (8), instead of (myArray.length)(9). Because in the for statement we use a < symbol to compare, if the bound is 8, the largest value we will reach will be 7, because 7 is the largest value less than 8. For this reason the 9th term in myArray (the term at the 8 index) will not be carried over, and the 9th term in newArray will be 0. The rest of the values in newArray will correspond with the values in myArray. When Printed in reverse, The for loop correctly starts at newArray.length-1

(8), and then ends at (0), so all of the values of newArray will be printed, (in reverse order). It also ends correctly at 0, by using a >= sign, showing that the loop will include 0, the first term. For this reason D is the correct answer, because the last term in newArray is zero, and the rest are the same as myArray, and it is printed out in reverse order. A is wrong because it does not account for the last value of newArray being 0. B and C are wrong because they contend that the last for loop's bounds either cut out the 0th or 9th term, which they do not.

4. Topic: Binary, Hex, Octal

Since we as humans have 10 fingers, we like to do arithmetic in base 10 numbers. Suppose we encounter 3 alien species who have 2, 8, and 16 fingers and they do arithmetic in base 2, 8, and 16 respectively. Suppose the 157 in base 10 bears some significance and we have to convert it into each species' base. What would the appropriate conversion be?

A.) 10011101, 235, 9D
B.) 10111001, 532, D9
C.) 10011101, 235, 913
D.) 10111001, 532, 913

Solution:

$$\begin{array}{r} 9 \\ 16\overline{)157} \end{array}\ r\ 13$$

$$\begin{array}{r} 0 \\ 16\overline{)9} \end{array}\ r\ 9$$

$$157_{10} = 9D_{16}$$

$$9D \to 1001 \leftrightarrow 1101$$

$$9D_{16} = 10011101_2$$

$$10|011|101 \to 2\ 3\ 5$$

$$10011101_2 = 235_8$$

$$157_{10} = 9D_{16} = 10011101_2 = 235_8$$

5. Topic: Enumeration

```
enum Direction
{
   Up, Down, Left, Right
}

public class ConnectFour
{
   public static void main(String[] args)
   {
       Direction[] moves = new Direction[4];
       moves[3] = Direction.Left;
       moves[2] = Direction.Up;
       moves[1] = Direction.Right;
       moves[0] = Direction.Down;

       if(moves[0] == Direction.Up)
           System.out.println(Direction.Up);
       else if(moves[3] == Direction.Left)
           System.out.println("2");
       else if(moves[2] == Direction.Up)
           System.out.println(Direction.Down);
       else
           System.out.println(" 1");
   }
}
```

What’s the output?

a. Down 1
b. Down
[✓ c. 2]
d. 2Down 1

Solution: c) 2
move[0] == Direction.Down and so it skips the first if statement, moves[3] == Direction.Left, so the next if statement is true, so the code runs System.out.println("2"); Finally the other if statement and else statement don’t run because in a chain of if else statements, when one is true, it doesn’t check the if statements after it.

Free Response Questions:

1. Topic: Arrays

Write a method with declaration to reverse the given array..

Solution:

```
public static void reverse(int[] array){
    for(int i = 0; i < array.length / 2; i++){
        int temp = array[i];
        array[i] = array[array.length - i - 1];
        array[array.length - i - 1] = temp;
    }
}
```

2. Topic: Arrays

What is the output of the following program? Show your work.

```
public static void main(String[] args) {
      int[] myArrA = new int[4];
      int myArrB[] = new int[]{5,6,7,8};
      int[] myArrC;

      myArrC = myArrA;

      for (int i = 0; i < myArrA.length; ++i)
            myArrA[i] = i+1;

      for (int j = 0; j < myArrA.length; j++)
            myArrC[j] += (j % 2 == 0) ? myArrB[j] - myArrA[j] : myArrB[j] +
            myArrA[j];

      for (int i : myArrC)
            System.out.print(i+" ");
}
```

Solution:

This problem is a little more involved, but if you take it step by step it’s actually not that bad. First we note the arrays in use for the problem. Initially, myArrA starts with size 4, with all elements 0 - [0,0,0,0] (as that is the default value for ints in Java). Next myArrB is directly initialized with four int values [5,6,7,8] accordingly.

Continuing on we have myArrC is declared as a pointer to an array of ints. Here it is important to note myArrC is a reference to an array of ints. Until it is initialized, it cannot be used. However, since we set myArrC equal to myArrA. This essentially means that any mention of myArrC is the same thing as a reference to myArrA. This fact is important considering the next two lines of code initialize the data in myArrA and myArrC, making the data [1,2,3,4].

The next lines of code is where the tricky portion lies. Essentially we’re looping through from index 0 to 3 (since the length of myArrA is 4), and changing the data that myArrC / myArrA is pointing to. Starting with j = 0, we note that the original value of myArrC[0] is 1. Next we evaluate the ternary operator to decide what gets added to it. We know 0 % 2 is 0, so we use the first condition, calculating that myArrB[0] = 5 and myArrA[0] = 1 (Note that this is the same value as myArrC[0] because both reference the same data). Finally we simply complete the operation myArrC[0] = 1 + 5 - 1, which evaluates to 5.

As we keep going, we update j = 1, and we go through the loop again, noting the values accordingly (myArrC[1] = 2, myArrB[1] = 6, myArrA[1] = 2). Here we know 1 % 2 equals 1, so we use the second condition, so the final calculation is myArrC[1] = 2 + 6 + 2, which equals 10. For j = 2, we get myArrC[2] = 3 + 7 - 3 which is 7 and for j = 3 we get myArrC[3] = 4 + 8 + 4 which is 16. So the final result printed out is “5 10 7 16”.

3. Topic : Method Overloading

A. What are the parts of a method signature? What can you change in the signature to overload a method?
B. What is the output of this program if it compiles?

```
public class Corgi{
        private String name;
        public Corgi(String name){
                this.name = name;
        }

        public void hungry(boolean isHungry){
                if(isHungry == true){
                        System.out.println(“Feed me”);
                }
                else{
                        System.out.println(“I am floofy”);
                }
        }
        public void hungry(boolean isHungry, int numTreats){
                if(isHungry == true && numTreats > 0){
                        System.out.println(“Hi I would like “ + numTreats + “ treats pls”);
                }
                else {
                        System.out.println(“I am not hungry for treats”);
        }
    }
    public int hungry(boolean isHungry(boolean isHungry, int numTreats)
        if(isHungry == true && numTreats > 0){
            return numTreats--;
        }
        else {
            return ++numTreats;
        }
}


public static void main(String[] args){
    Corgi ned = new Corgi("Ned");
    ned.hungry(true,5);
}
```

C. If the last version of the hungry() method above was deleted, what would the output of the program be?

Solution:

A. Method Name, Number Of Arguments, Types Of Arguments and Order Of Arguments; Method overloading allows you to change the number of arguments, types of arguments, and the order of arguments.

B. The program will not compile since two methods have the same number, order, and type of arguments and only the return type is changed. This causes a compiler error since it does not know which version of the method to call.

C. The output to the program would be "Hi I would like 5 treats pls"

4. Topic: pass primitive types vs. referenced types

```
public class Student {
    public int grade;

    public Student(int g) {
        grade = g;
    }

    public static void increaseGrades(int[] grades, Student student, int
grade) {
        grades[0] += 25;
        student.grade = 100;
        grade = 101;
    }

    public static void main(String[] args) {
        int[] grades = {85, 80, 90};
        Student bob = new Student(grades[1]);
        int gradeX  = grades[2];
        increaseGrades(grades, bob, gradeX);
        System.out.println(grades[0] + bob.grade + gradeX);
    }
}
```

Solution: 300

Explanation:

grades is an array, and the variable itself holds a reference to the memory allocated on the heap. *bob* is an instance of the Student class, and the variable itself holds a reference to the data on the heap. *gradeX* is a primitive type variable and its data is held on the stack.

When the three variables are passed into the function increaseGrades as parameters, they are all copied. The reference to the memory where *grades* (the array) and *bob* (the instance of a Class) are located is copied to the parameter and *gradeX*'s primitive data is copied to the parameter variable. Since *gradeX* is copied to *grade* and has no reference to the original memory, no change occurs to the *gradeX* variable in the main method. Even though the *grades* and *bob* variables references are copied to the parameter variables, they still reference the same memory on the heap. Therefore, any changes that occur to the data inside the increaseGrades method will persist outside of that method.

Note: There is a caveat to the last sentence. We can change the data that is referenced in the *grades* and *student* variables, and those changes will persist. However, in increaseGrades, if we reassign *grades* (grades = new int[3]) or *student* (student = new Student(100);) that change

will allocate new memory on the heap and those variables will no longer reference the original memory.
For more info: https://www.geeksforgeeks.org/g-fact-31-java-is-strictly-pass-by-value/

5. Topic: Classes, specifically static variables and constructors

Given the following class definition,

```
public class Hero {
      String name;
      int power;
      static int heroCount = 0;

      public Hero(String name, int power) {
            this.name = name;
            this.power = power;
            heroCount++;
      }

      static void death(int deathCount) {
            heroCount -= deathCount;
      }
}
```

What is the output of the following code snippet?
Assume that the main method is implemented and ran properly

```
public static void main(String args[]) {
      Hero jim = new Hero(“jim”, 12);
      Hero pam = new Hero(“pam”, 9);
      Hero andy = new Hero(“andy”, 4);
      jim.heroCount--;

      Hero erin = new Hero(“erin”, 10);
      Hero michael = new Hero(“michael”, 7);

      erin.death(2);

      System.out.println(Hero.heroCount);
}
```

Solution:
The console will output 2. The idea here is that static variables are shared amongst the class and all of its objects. Any time you access a static variable through one object, it affects everything. Students also have to understand that a constructor is run upon the creation of the object. Thus, every time a Hero object is made, 1 is added to heroCount.

6. Topic: Staggered Arrays

Given the following program

```
public class Main {

    public static void main(String[] args) {
          int[][] arr1 = new int[15][0];
          int[][] arr2 = arr1;
          arr1[0] = new int[] {6,7,8,9,10,11,12,13};    //Line 3
          arr2[0] = new int[]{ 1,2,3,4,5};            //Line 4
          arr1[1] = arr2[2] = arr1[0];

          arr1[1] = new int[] {-1,-2,-3,-4, -5 };           //Line 6

          arr2[2][0] = arr1[1][3];
          arr2[2][4] = arr2[1][4];

         for (int i = 0; i < arr2[2].length; i++) {
                  arr2[2][i] *= 2;
                  System.out.println(arr2[2][i]);
               }
    }
}
```

What is the output of the following code snippet?

Solution:
-8 4 6 8 -10

This problem tests a students understanding of staggered arrays and how arrays are stored in memory and accessing them. arr1 is first initialized to a 2d array with 15 rows and then arr2 is referenced to arr1, so when Line3 is initialized Line4 overwrites that because it references the same array. Arr1[1] and arr2[2] will both equal to {1,2,3,4,5}. arr1[1] also equals arr2[1] in line 6.

arr[2][0] = -4 and arr[2][4] = -5

7. Topic: Classes

Design a Bank Account class. A bank account will contain a double to hold the remaining balance and an email (String).
If a bank account is created without an email, the default balance will be equal to 100.00. If an email is provided while creating a bank account, the balance will be equal to 1.5 times the length of the email.
Additionally, all bank accounts will have a function call account Information that will print the account balance if the email was provided; otherwise, it will print "Inactive account."

Example 1 output (Default bank account, no email provided)
Inactive account.

Example 2 output (email = "abc@ufl.edu")
Balance: 16.5

Solution:

```
public class BankAccount {
  double balance;
  String email;

  public BankAccount() {
    balance = 100.0;
  }

  public BankAccount(String email) {
    this.email = email;
    balance = 1.5 * email.length();
  }

  public void accountInformation() {
    if(email == null) {
      System.out.println("Inactive account.");
    }
    else {
      System.out.println("Balance: "+ balance);
    }
  }
}
```

8. Topic: Strings

```
public class Test {

    public static void main(String[] args) {
        String output = "";
        String master = "The-Quick-Brown-Fox-Jumped-Over-The-Lazy-Dog";
        String temp = master.substring(4, 9);
        output = temp + master.substring(31, 44);

         if(master.substring(0,3).equals("the")){
            output+= " true";
         }
         else{
             output+= " false";
         }

        System.out.println(output);

    }
}
```

What will the print statement print out?
Quick-The-Lazy-Dog false

What could you change in the italicized line in order to make the statement true?
if(master.substring(0,3).equalsIgnoreCase("the"))
Or
if(master.substring(0,3).equals("The")){

9. Topic: Strings

A palindrome is a word that stays the same after you reverse it. "racecar", "noon", and "aibohphobia" (the fear of palindromes), are examples of palindromes.

Write a method "isPalindrome" that takes a single-word String as a parameter, and returns a boolean value. isPalindrome should return ***true*** if the String parameter is a palindrome, and ***false*** if it isn't a palindrome. *You can assume that the String parameter will have lowercase characters only. The String parameter can be of any length, including 0. Strings of length 0 and 1 are to be considered palindromes in this case.

Sample Solution:

```
public static boolean isPalindrome(String word)
{
    /* used to iterate through the string
    in both directions at once. */
    int left, right;

    /* word.length() is even, so there are two middle indices */
    if(word.length() % 2 == 0) {
        // truncates to lower of two middle indices
        left = word.length()/2 - 1;
        right = left+1; // upper middle index
    }
    /* word.length() is odd, so only one middle index */
    else {
        left = word.length()/2;
        right = left;
    }
    /* iterate in both directions from the middle, checking if characters are
equal */
    while(left >= 0 && right < word.length()) {
        // if they aren't equal, word isn't a palindrome
        if(word.charAt(left) != word.charAt(right))
            return false;
        left--;
        right++;
    }
    return true;
}
```

Alternative Solution with checking the length of the string:

```
public static boolean isPalindrome(String word) {
      // counter variables pointing to the beginning
      // and the end of the string
      int i = 0, j = word.length() - 1;

      // as long as there are characters to compare
      while (i < j) {
         // if there is a mismatch
         if (word.charAt(i) != word.charAt(j))
            return false;
         // move on to the next pair of characters
         i++;
         j--;
      }
      return true;
}
```

Additional Review Questions:

1. Topic: Arrays

```
public static void main(String[] args) {

   String[] arr1 = {"", "", "", "", ""};
   String[] arrayTwo = new String[5];

   System.out.println(arr1 == arrayTwo);
   System.out.println(arr1.equals(arrayTwo));


   double[] arr3 = {0.0, 0.0, 0.0, 0.0, 0.0};
   double[] arrayFour = new double[5];
   System.out.println(arr3 == arrayFour);

}
```

What is the output of the given code?

a. False
   False
   true
b. true
   True
   true

c. false
False
false

d. true
False
true

Explanation:

When just using ==, Java compares the memory addresses of the two arrays. Using the new keyword creates a new block of memory that the regular array doesn't. Also, when creating a new array, java will put default values in the array.

2. Topic: Two-Dimensional Arrays

Which of the following is legal Java code?

a) char[][] connectFour;

b) int[][] matrix = new int[][];

c) char[][] document = new char[][72];

Solution:

a) You can declare multi-dimensional arrays without specifying the dimensions

b) This will generate a compile error

c) If you declare a multi-dimensional array specifying only one dimension,
It must be the first dimension that is specified. First dimension specification is mandatory. The second dimension is optional

3. Topic: Classes

Given a 2D Cartesian coordinate system, the Point in the coordinate is defined as following:

```
class Point {
    int x, y;
    public Point(int x, int y) {
        this.x = x;
        this.y = y;
    }
}
```

You are required to implement a series of tools to process the given points.
1. **Find the middle of any two given points**. The definition for midPoint of two points *A(x1,y1) and B(x2,y2) is: x_mid=(x1+x2)/2, y_mid=(y1+y2)/2*. If the x,y value of the midPoint is not integer value, ROUND IT UP to the next integer which is larger or equal to the original value. *E.g. (2,5, 2.6)=>(3,3), (-1.1,0)=>(-1,0)*
2. **Calculate the Euclidean distance between any two given points**. In Cartesian coordinates, the Euclidean distance between the points *A(x1,y1)* and *B(x2,y2)* is defined as *((x1-x2)^2 + (y1-y2)^2)^(½)*.
You are NOT allowed to modify the Point class and the signature of the starter code. The starter code for the Tool class:

```
public class Tool {
    public static Point midPointOf(Point p1, Point p2) {
        // your code
    }

    public static double euclideanDistanceOf(Point p1, Point p2) {
        // your code
    }
}
```

Solution:

```
public class Tool {
    public static Point midPointOf(Point p1, Point p2) {
        return new Point((int) (Math.ceil((p1.x + p2.x) / 2.0)),
                    (int) (Math.ceil((p1.y + p2.y) / 2.0)));
    }

    public static double euclideanDistanceOf(Point p1, Point p2) {
        return Math.sqrt((p1.x - p2.x) * (p1.x - p2.x)
                        + (p1.y - p2.y) * (p1.y - p2.y));
    }
}
```