# **CIS 36A :: LAB 07 - Advanced Classes and Methods**

#### **Student Name: Esmatullah Nickzad**

### **Task 1: Definitions & Concepts**

**Instructions: Briefly** answer the questions below.

1. Keywords: To your best knowledge, describe below words:
   1. Public => Answer: its referred to members of a class that can be accessed from anywhere in the program.
   2. Private => Answer: its accessible only within the class , they are defined in .
   3. static => Answer: A static member belongs to the class it self , not to any objects of the class
   4. Varargs => Answer: it means variable arguments “ varags “ is a feature that allows a method to accept a variable number of arguments.
   5. Argument => Answer: An argument is a value passed to a method when its called , methods can have zero or more arguments.
2. What is *method overloading*?  
   => Answer: method overloading lets you have multiple methods with the same name in a class, but they must have different parameter lists. This promotes code readability and reusability.
3. What is an inner class?  
   => Answer: an inner class is a class defined within another class. It has special access privileges.

Inner classes can access the private members of the outer class

They are typically used to group related functionality together with the outer class, improving code organization.

1. Why might you need to use a static method?  
   =>Answer: for two main reason: utility functions : when a method operates on general input and doesn’t rely on the specific state of an object.

Shared state: if a method works with data share by all instances of a class , making it static ensures all objects access the same data.

1. To make a member of class accessible by only other members of its class, what access modifier must be used?  
   => Answer: private access modifier

### **Task 2: Understanding Programming**

Instructions: Answer each question below.

**Exercise 1:**

Given this fragment,

class x {  
 private int count  
 }

is the following fragment correct?   
  
class Y {  
 public static void main(String [] args) {  
 X ob new X();  
 Ob.count = 10;

|  |
| --- |
| Explain: it is not correct; there are some syntax errors;  Class ‘x’ is not capitalized consistently throughout the code, it should be capitalized to match the class name definition.  The variable ‘count’ is declared a private in class ‘X’ it means it can not be directly accessed from outside the class,  In the main method class Y , there is a typo in the object creation statement. It should be ‘X’ ob = new X(); instead of ‘X’ ob new x();,  And lastly in the statement Ob.count =10 ,’Ob’ should be capitalized as ‘ob’ to match the variable declaration. |

**Exercise 3:**

Given this class,

class Test{

int a;

Test(int i) { a = i; }

Write a method called **swap()** that exchanges the contents of the objects referred to by two Test object references.

|  |
| --- |
| Answer:  Class Structure:  We define a class named Test with two private integer variables, val1 and val2.  A constructor initializes these variables when a Test object is created.  swap() Method:  This method is named swap and is public, allowing other objects to call it.  It takes another Test object named other as a parameter.  Swapping Values:  Inside the swap method, a temporary variable temp of type int is declared.  The value of this.val1 (which refers to the val1 of the current object) is assigned to temp.  The value of other.val1 (which refers to the val1 of the other object) is assigned to this.val1. This effectively replaces the val1 of the current object with the val1 of the other object.  Finally, the value stored in temp (which was originally this.val1) is assigned to other.val2. This replaces the val2 of the other object with the original val1 of the current object. as you see on the below:  class Test {  private int val1;  private int val2;  public Test(int value1, int value2) {  this.val1 = value1;  this.val2 = value2;  }  public void swap(Test other) {  // Swap values between this and other object  int temp = this.val1;  this.val1 = other.val1;  other.val2 = temp;  }  } |

**Exercise 4:**

Is the following fragment correct?

class X {

int meth (int a, int b) { … }

String meth (int a, int b) { … }

|  |
| --- |
| **Explain:**  No, it is not correct because it violates the rule of method overloading in java, method overloading allows multiple methods with the same name but different parameter types or number of parameters. In this fragment, both methods have the same name ‘ meth’ and the same parameter types ‘ int a, int b’ but they differ in return types ‘ int ‘ and ‘String’ in java, methods cannot be overloaded based solely on difference in return types; the compiler will generate an error due to ambiguity. |

Exercise 5: Write an **iterative** method that displays the content of a string backward.

|  |
| --- |
| A screen shot of a computer  Description automatically generated |

Exercise 12: Create a “varargs” method called **sum()** that sums the int values passed to it. Have it return the result. Demonstrate its use.

|  |
| --- |
| A screen shot of a computer program  Description automatically generated |

Exercise 19: The method **hasDuplicateValues( )** shown below is supposed to return **true** if the array contains any repeated integers and return **false** if all integers in the array are unique. However, it doesn't work correctly. Explain why it is wrong and then fix it.

Answer: The method hasDuplicateValues () is incorrect because it compares each elemnt of the array with every other element , including itself, which leads to incorrect results, and also it lacks proper scoping for the inner loop

To fix this issue: we need to adjust the inner loop to start from the element after the current element and properly use curly braces to define the scope of the inner loop. Here is the correct one in below:

boolean hasDuplicateValues(int[] data) {

for (int i = 0; i < data.length; i++) {

for (int j = i + 1; j < data.length; j++) {

if (data[i] == data[j]) {

return true;

}

}

}

return false;

}

Exercise 23: Suppose you have a method that takes as its parameters two integer arrays **data1** and **data2** of the same length and it copies all the data from **data1** to **data2** and then clears **data1** (that is, it sets all the values in **data1** to 0). What would happen if someone invoked this method and passed in the same array for both arguments rather than passing in two distinct arrays?

Answer:

If someone invokes this method and passes in the same array for both arguments instead of passing in two distinct arrays, both arrays would end up containing all zeros after the method execution. This is because arrays in Java are passed by reference, so both data1 and data2 would refer to the same array. so, when the method clears data1, it's actually modifying the same array referenced by data2 as well.  
here I have worked in an **example;**

public class ArrayManipulation {

public static void copyAndClear(int[] data1, int[] data2) {

for (int i = 0; i < data1.length; i++) {

data2[i] = data1[i];

}

for (int i = 0; i < data1.length; i++) {

data1[i] = 0;

}

}

public static void main(String[] args) {

int[] array = {1, 2, 3, 4, 5};

copyAndClear(array, array);

System.out.println("data1: " + java.util.Arrays.toString(array));

System.out.println("data2: " + java.util.Arrays.toString(array));

}

}

**Exercise 24:** Suppose that a class has an overloaded method named **add** with the following two implementations:

double add(int x, double y) { return x + y; }

double add(double x, int y) { return x + y + 1; }

What, if anything will be returned by the following method calls?

1. add(3, 3.14)
2. add(3.14, 3)
3. add(3, 3)
4. add(3.14, 3.14)

**Answer: I will explain each one by one:**

1. add (3, 3.14)

The method add(int x, double y ) is applicable in here,

It returns 3+3.14 = 6.14.

1. add(3.14, 3)

The method add(double x, int y) is applicable in here

It returns 3.14+3+1 = 7.14.

1. add (3,3)

Both methods are applicable since the first argument can be promoted to either int or double, it will choose add(intx, int double)

It returns 3+3=6.0.

1. add(3.14, 3.14)

Neither method directly matches the argument types. However both arguments can be promoted to double , since there is no clear choice java will face a compilation error due to ambiguity.

Exercise 33: The following code will not compile. Explain what is wrong.

class Oops {

int x = 3;

static void changeX() { x = 4; }

}

**Answer:**the problem is that it attempt to modify a static variable from a non static method.

The variable x is declared as static that means there is only one copy of x shared by all instances of the oops class. There is two ways to fix these issues:

1. modify , change x to be static
2. create an instance and modify its variable.

Exercise 37: What is the difference between the following two methods? More precisely, the bodies of the methods are the same, but the parameters aren't. What does the vararg version **addUp1** allow us to do differently than the array version **addUp2**?

int addUp1(int ... v) {

int sum = 0;

for(int x : v)

sum += x;

return sum;

}

int addUp2(int[] v) {

int sum = 0;

for(int x : v)

sum += x;

return sum;

}

**Answer:**

The two methods, addUp1 and addUp2, are similar in functionality but difference in the way they accept arguments:

addUp1(int ... v):

This method uses varargs, denoted by the ellipsis (...) after the type int.

It allows the caller to pass a variable number of integer arguments directly to the method, without explicitly creating an array.

For example, you can call this method with addUp1(1, 2, 3) and addUp1(1, 2, 3, 4, 5) without explicitly creating an array.

addUp2(int[] v):

This method accepts an array of integers as its argument.

It requires the caller to explicitly create an array and pass it to the method.

For example, you need to create an array before calling this method, such as int[] array = {1, 2, 3}; and then call addUp2(array);.

In summary, the varargs version addUp1 allows the caller to pass a variable number of arguments directly to the method without creating an array, whereas the array version addUp2 requires the caller to explicitly create and pass an array of integers.

### **Task 3: Programming Exercises**

Instructions: Use any IDE to write and execute below exercises from the book chapter 6. Attach Snipping photos of your source code and execution of the code in the console. Make sure to create separate files for each exercise.

Exercise 16: Implement a **string2charArray( )** method that takes a **String** as its parameter. It creates and returns a **char** array containing the same characters in the same order as in the string.

A screen shot of a computer program

Description automatically generated

Exercise 17: Implement a **charArray2string( )** method that takes a **char** array as its parameter. It creates and returns a **String** that contains the same characters in the same order as in the array.

A screen shot of a computer program

Description automatically generated

Exercise 30 - Implement an **equalArrays( )** method that takes two integer arrays as parameters and returns **true** if both arrays are the same length and have equal values at corresponding indices.

A: Iteratively

A computer screen shot of a program

Description automatically generated

Exercise 31 - A: Write a method **reverse( )** that takes an integer array as its parameter and reverses the order of the elements in the array.

A: Iteratively

A screen shot of a computer program

Description automatically generated

**Task 4 - Programming Application**

A rational number is a number that can be represented as the ratio of two integers. For example, 2/3 is a rational number, and you can think of 7 as a rational number with an implicit 1 in the denominator or 7/1.

**Continue from Lab 4:**

1. Make the instance variables private and create a proper accessor and mutator methods to get and set the instance variables.
2. Write three **constructors** that;
   1. one takes two arguments and sets the numerator and denominator with these two parameters. This constructor should call the reduce method to always reduce the rational to its simplest form.
   2. one takes one argument and sets the numerator to this number and denominator to 1. This constructor should call the first constructor with two arguments, using *this*
   3. one that takes no argument and sets both numerator and denominator to 1. This constructor should call the first constructor with two arguments, using *this*
3. Write a main method that creates a new object with type Rational, sets its instance variables to some values, and displays the object.

A screen shot of a computer program

Description automatically generated