UBC OKANAGAN

COSC 121
Computer Programming II

Final Exam Revision

(Cumulative)

Dr. Mostafa Mohamed

Acknowledgement: The slides mainly rely on the textbook of Y. D. Liang titled “Introduction to Java Programming, 10th Ed.”, Pearson Edu. Inc.

COSC 121 Final Exam

Total marks: **35% of the course grade**

Time limit: **2.5 hours**

The exam covers **all course materials** as indicated in the syllabus and during the lectures.

In order to pass the course, you must get at least:

- 50% of the weighted sum of the final + midterm exams, and
- 50% of the total grade

Final Exam: Things to Bring

Required:

- UBC ID card

Allowed:

- Pencils, pens, eraser, and sharpener.
- Using back of exam sheets for your answers if indicated clearly.
- Ask questions only in cases of supposed errors or ambiguities in examination questions

Not allowed:

- Books, notes, or electronics (calculators, computers, sound or image players/recorders/transmitters (including telephones)).
- Extra sheets of paper. Use backs of exam sheets for scrap paper.
- Entering the exam room after the expiration of one half-hour from the scheduled starting time.
- Leaving during the first half-hour of the exam

Final Exam Structure

Multiple choice questions

Short answer questions

- Analysis questions
  - Answer questions related to some code (what will happen if…, what is the value of…, replace a piece of code, what is the output, …etc.)
- Essay questions (“Briefly explain..”).
  - Whenever applicable, use an illustration to support your answer.

Long answer questions.

- both short and long coding questions.
- write a program from scratch
- given partial code, modify or add more code in order to implement missing functionality (classes, methods, etc.).

May have: Bonus question (coding)

Cheat Sheet

I will **NOT** provide a cheat sheet in the exam.

However, you **CAN** bring yours. The rules are:

- You can use up to **TWO**-side of an A4 paper as the cheat sheet.
- Your notes can be handwritten or printed.
- Your NAME must be written on your cheat sheet.
- You can only bring one cheat sheet.
- Switching cheat sheets with others during the exam is not allowed.
- No other aids are allowed (e.g. electronics, etc). The exam is closed book

Contents

What you have learned in COSC121

OOP basics

- Class structure, attributes/methods/constructors
- Visibility modifiers, `static`, `final`, `this`, `super`
- Reference variables vs. primitive variables
- The `Object` class
  - Methods: `toString()`, `equals()`, `clone()`
- Array of objects

OOP Pillars

- 1. Encapsulation
- 2. Inheritance
- 3. Polymorphism
  - Reference of supertype referring to different subtypes
    - Array of objects of different but related types
  - Polymorphism Rules
    - can only access class members of the reference type, dynamic/static binding

`instanceof` operator

Object Casting

What you have learned in COSC121

Abstract classes / Interfaces

- Abstract methods
  - common behaviour but different implementation
  - in abstract classes or interfaces
  - must be implemented by concrete subclasses / implementing classes
- Can create references of an abstract class / interface type, but NOT objects

Java standard interfaces such as:

- `Comparable<T>`
- `Comparator<T>`
- `Cloneable`
  - shallow / deep clone
- `Serializable` (for Binary I/O)
- ~~Listener interfaces (for event driven programming)~~

What you have learned in COSC121

Exception handling

- Types of exceptions (Checked vs. Unchecked)
- Techniques of dealing with exceptions
  - Declare in method header (`throws`), or
  - Handle within method (`try..catch`)
- Exception propagation
- Defensive Programming vs. Exception Handling

I/O Streams:

- Steps for using I/O streams
- Text I/O
  - Text I/O classes and their methods
    - How to use with standard I/O, files, and web.
    - `Scanner, PrintWriter, BufferedReader, BufferedWriter`
  - The `File` class
  - Try-with-resources

What you have learned in COSC121

I/O Streams (cont'd)

- Binary I/O
  - Reading/writing using binary I/O classes and methods
    - `[File|Data|Object][Input|Ouput]Stream`
    - `Buffered[Input|Output]Stream`
      - Wraps around `File[Input|Output]Stream`
    - Reading complete file, how to check the end of file
    - Appending data to file using `FileOutputStream` (with or w/t PrintWriter)
  - The `Serializable` Interface
  - The `transient` Keyword

Recursion

- When to use recursion vs. iteration.
- How to use recursion
  - With or w/t recursive helper methods
- What is tail-recursive methods?

What you have learned in COSC121

Java Collections & Intro to Generics

- Collections vs. Arrays
- Intro to Generics
  - Also *autoboxing* (`int` → `Integer`, `double` → `Double`, etc)
- Using collection classes (and their methods)
  - `ArrayList`, `LinkedList`, `Stack`, `Queue`, `PriorityQueue`
- Implementing collection classes
  - `ArrayList`, `LinkedList`, `Stack`, `Queue`
- Useful methods from Collections and Arrays classes
  - Array ←to→ List

    list= Arrays.asList(array) list.toArray(array)
  - Collections' `sort()`, `max()`, `min()`, `shuffle()`

Sorting

~~Lambda Expression~~

~~Basics of Networking in Java (Client-Server)~~

~~GUI~~

You should be familiar with the main concepts covered in COSC111/123.
e.g., variables, data types, operators, loops, selection, methods, and arrays.

Any specific parts you want to review?

Any specific questions ?

Example of MCQ or Analysis Questions

Which class can access which field?

package p1

```
public class C1 {
  public int x;
  protected int y;
  int z;
  private int u;

  protected void m(){}
}
```

```
public class C2 {
  C1 o = new C1();
  can access o.x;
  can access o.y;
  can access o.z;
  cannot access o.u;

  can invoke o.m();
}
```

```
public class C3 extends C1
{
  can access x;
  can access y;
  can access z;
  cannot access u;

  can invoke m();
}
```

package p2

```
public class C4 extends C1
{
  can access x;
  can access y;
  cannot access z;
  cannot access u;

  can invoke m();
}
```

```
public class C5 {
  C1 o = new C1();
  can access o.x;
  cannot access o.y;
  cannot access o.z;
  cannot access o.u;

  cannot invoke o.m();
}
```

Example of MCQ or Analysis Questions

Polymorphism and Dynamic Binding

```
public class Person {
    public void method1(){System.out.println("This is person 1.");}
    public void method2(){System.out.println("This is person 2.");}
}
```

```
public class Man extends Person{
    public void method1(){System.out.println("This is a man.");}
}
```

```
public class Woman extends Person{
    public void method1(){System.out.println("This is a woman.");}
}
```

```
public class DynamicBindingTest {
    public static void main(String[] args) {
        Person p1 = new Person();
        Person p2 = new Man();
        Person p3 = new Woman();
        p1.method1();
        p2.method1();
        p3.method1();
        p1.method2();
        p2.method2();
        p3.method2();
    }
}
```

What is the output?

```
This is person 1.
This is a man.
This is a woman.
This is person 2.
This is person 2.
This is person 2.
```

Example of MCQ or Analysis Questions

Method Matching vs. Dynamic Binding

```
public class DynamicBinding2 {
    public static void main(String[] args) {
        Object j1 = new GradStudent();
        Object j2 = new Student();
        Object j3 = new Human();
        Object j4 = new Object();
        m(j1);
        m(j2);
        m(j3);
        m(j4);
    }
    private static void m(Object object) {
        System.out.println(object.toString());
    }
    private static void m(Student s){
        System.out.println("No dynamic binding");
    }
}
class Human{
    public String toString(){return "Human";}
}
class Student extends Human{
    public String toString(){return "Student";}
}
class GradStudent extends Student{
}
```

What is the output?

```
Student
Student
Human
java.lang.Object@5b2b6037
```

Explain how we got this output?

Example of MCQ or Analysis Questions

Which of the following classes defines a legal abstract class?

```
class A {
  abstract void unfinished() {
  }
}
```

(a)

```
public class abstract A {
  abstract void unfinished();
}
```

(b)

```
class A {
  abstract void unfinished();
}
```

(c)

```
abstract class A {
  protected void unfinished();
}
```

(d)

```
abstract class A {
  abstract void unfinished();
}
```

(e)

```
abstract class A {
  abstract int unfinished();
}
```

(f)

Answer: D, E, F are legal abstract classes

Others are illegal: A and C are not abstract classes, and B is illegal declaration

Example of MCQ or Analysis Questions

Which of the following interface definitions is valid and why?

a)
```
public interface interface1{
    void displayHello(){
        System.out.println("Hello");
    }
}
```

b)
```
public interface interface2 {}
```

c)
```
public interface interface3{
        int MAX = 200;
}
```

d)
```
public interface interface4{
        final int MAX = 200;
        String text;
        void display();
}
```

e)
```
public interface interface5{
        boolean display(int a);
}
```

(a)Invalid – because it contains a concrete method.

(b)Valid – although perhaps not very useful.

(c)Valid – this defines a constant as the modifiers public static final are implicitly applied.

(d)Invalid – because text is either an attempt at an instance variable or is an uninitialized constant.

(e)Valid – this defines a single method, which is implicitly defined as public abstract.

Example of MCQ or Analysis Questions

```
public class Robot {
    int x = 10, y = 20;
    transient String location = "Canada";
    public void amethod(){}
}
```

What is wrong ?

Can't write Robot instance to file until it is Serializable

```
import java.io.*;
public class TestObjStream {
    public static void main(String[] args) throws Exception {
        // WRITE student test scores to the file
        ObjectOutputStream out =
                new ObjectOutputStream(new FileOutputStream("temp.dat"));
        out.writeUTF("My Robot"); out.writeObject(new Robot());
        out.close();
        // READ: must use the SAME ORDER
        ObjectInputStream in =
                new ObjectInputStream(new FileInputStream("temp.dat"));
        String s = in.readUTF();
        Robot r = (Robot)in.readObject();
        System.out.println(s + ", " + r.location + ", (" + r.x + ", " + r.y + ")");
        in.close();
    }
}
```

Example of MCQ or Analysis Questions

```
import java.io.Serializable;
public class Robot implements Serializable{
    int x = 10, y = 20;
    transient String location = "Canada";
    public void amethod(){}
}
```

What is the output?

```
My Robot, null, (10, 20)
```

```
import java.io.*;
public class TestObjStream {
    public static void main(String[] args) throws Exception {
        // WRITE student test scores to the file
        ObjectOutputStream out =
                new ObjectOutputStream(new FileOutputStream("temp.dat"));
        out.writeUTF("My Robot"); out.writeObject(new Robot());
        out.close();
        // READ: must use the SAME ORDER
        ObjectInputStream in =
                new ObjectInputStream(new FileInputStream("temp.dat"));
        String s = in.readUTF();
        Robot r = (Robot)in.readObject();
        System.out.println(s + ", " + r.location + ", (" + r.x + ", " + r.y + ")");
        in.close();
    }
}
```

Example of MCQ or Analysis Questions

What is wrong ?

```
import java.io.*;
public class Ex1 {
    public static void main(String[] args) {
        try (FileInputStream fis = new FileInputStream("test.dat");) {
        } catch (IOException ex) {
            ex.printStackTrace();
        }
         catch (FileNotFoundException ex) {
             ex.printStackTrace();
         }
    }
}
```

FileNotFoundException is unreachable

Example of MCQ or Analysis Questions

After the following program is finished, **how many bytes** are there in the file **t.dat**?

```
import java.io.*;
public class Ex4 {
    public static void main(String[] args) {
        try (DataOutputStream output = new DataOutputStream(
                new FileOutputStream("t.dat"));) {
            output.writeChar('A');
            output.writeUTF("DEF");
        } catch (Exception e){}
    }
}
```

7 bytes

- **2 for char**
- **5 for UTF = 2 for length + 1 for each letter**

Example of MCQ or Analysis Questions

What is the output? What does the program do?

Also, identify base cases and recursive calls.

```
public class Practice1 {
    public static void main(String[] args) {
        System.out.println("Sum: " + m1(5));
    }

    public static int m1(int n) {
        if (n == 1)
            return 1;
        else
            return n + m1(n - 1);
    }
}
```

o/p: 15

Example of MCQ or Analysis Questions

Tail-recursive or not? Explain

(a)

```
public static int factorial(int n){
    if(n == 0)                          //stopping condition
        return 1;
    else
        return n * factorial(n-1);  //recursive call
}
```

(b)

```
public static boolean isPalindrome(String s) {
    if (s.length() <= 1)                                    // Base case1
        return true;
    if (s.charAt(0) != s.charAt(s.length() - 1))    // Base case2
        return false;
    return isPalindrome(s.substring(1, s.length() - 1));
}
```

(a) **Not tail** , because there is a pending operation, namely multiplication, to be performed on return from each recursive call.

(b) **Tail**. No pending operations after the recursive call.

Example of Analysis / Short Answer

Which one of the following data structures is the best to use in a specific situation?

- Fixed length, a lot of setting and getting, but no insertion or deletion.
  - Fixed size arrays
- Unknown size, lots of adding to the end, and lots of getting by index
  - ArrayList

What is the worst case for a given sorting algorithm expressed in Big-O notation?

Apply the sorting algorithm X on a given array of elements.

- By hand while explaining all the steps.

Example of Analysis / Short Answer

Briefly explain

- How to create an array of objects of different type
  - hint: polymorphism
- Abstract classes vs. interfaces
- Exception propagation
- how to implement methods to add/remove/etc. elements to a collection,
- the different types of linked lists
- recursive helper methods (benefits?),
- tail-recursive methods vs. non-tail ones?
- a specific sorting algorithm,
- Why you would use one sorting algorithm over the other.

CODING QUESTIONS!

You will be asked to write code either

- **from scratch** or
- **based on given partial-program (**where you add extra code similar to the course project, i.e. the Teeny-Tiny Farm, or doubly-linkedlist assignment).

If you are given a partial program, you may be asked to:

- Add code to a given method to handle invalid input (e.g., using try..catch)
- Add a method to this class to read data from a binary file.
- Add a code to a custom list class to add/remove/get/set/etc. items in a given data structure.
- etc

Clicker Question

```
public class Human {
    public int age;
    Human(int ag){ age = ag;}
    Human(){ age = 10;}
}
```

A. The code defines no constructor

B. The code defines one constructor

C. The code defines two constructors

D. The code defines one method and one default constructor

E. None of the above

Clicker Question

`static` keyword in Java means

A. The variable cannot be modified

B. The variable can only be modified at runtime

C. The variable is associated with the class

D. The variable is associated with the object

E. None of the above

Clicker Question

public static void main(String[] args) means

A. this method can be run by anyone

B. this method returns nothing

C. this method is associated with the class

D. all of the above

E. A, and B but not C.

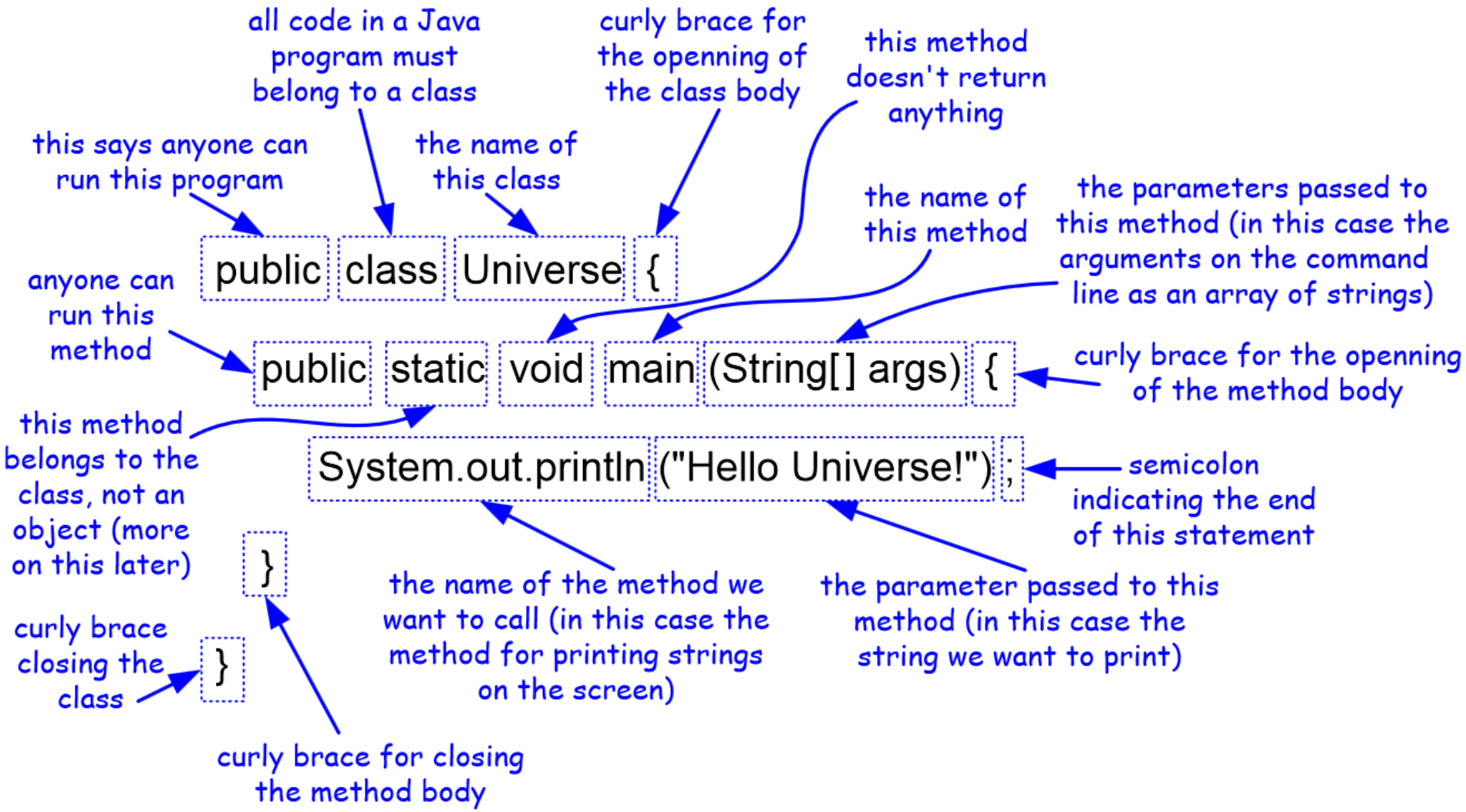


Figure 1.1: A "Hello Universe!" program.

http://higheredbcs.wiley.com/legacy/college/goodrich/0470383267/miss_fig/fig1.1.pdf

Clicker Question

The `final` keyword in Java means

A. The variable must be assigned an initial value in the same declaration statement, and it can never be assigned a new value later.

B. Once the variable is assigned an initial value, it cannot be changed

C. The variable can only be changed by an internal method

D. The variable cannot be inherited

final keyword

The **`final`** keyword in Java means

- A final class cannot be subclassed.
- A final method cannot be overridden by subclasses.
- A final variable can only be assigned once.

Clicker Question

What is the output?

```
public class Example {
    public static void main(String[] args){
      Gnome g = new Gnome();
      System.out.println(g.age);
      modify(g);
      System.out.println(g.age);
    }
    static void modify(Gnome g){
      g.age = 123;
    }
}
class Gnome {
    public int age;
    Gnome(){ age = 204;}
}
```

A. 123
   204

B. 204
   204

C. 123
   123

D. 204
   123

E. Error

Clicker Question

Inheritance is *merely* the capability of a class to use the properties and methods of another class.

How would you judge this definition?

A. EXCELLENT (complete answer)

B. GOOD (partial answer)

C. MEDIOCRE (totally unrelated)

http://home.cogeco.ca/~ve3ll/jatutor0.htm

Clicker Question

You need to implement 3 different types of ***trees***. All will have ***different*** implementations of the ***find***, ***insert***, and ***remove*** methods. How would you design your code?

A. Create 3 classes.

B. Create an abstract class and 3 child classes

C. Create a parent class and 3 child classes

D. Use cloning to avoid duplicating code

E. Create an interface implementing the default methods then a class implementing the interface

Clicker Question

```
public class Q {
  public static void main(String args[]){
     Cow cow = new Cow("Bolt");
     Dog dog = new Dog("Rover");
     Snake snake = new Snake("Ernie");
     Animal[] a = {cow, dog, snake};
     for (Animal animal : a) {animal.speak();}
}}
```

This code is an example of

A. Inheritance
B. Polymorphism
C. Overriding
D. Reflection
E. None of the above

Clicker Question

```
public abstract class Animal {

   private String name;

   public Animal(String nm) { name=nm; }

   public String getName() { return name; }

   public abstract void speak();

}
```

The code

A. is good

B. Should be implemented as an interface

C. generates an exception

D. is missing { } after speak()

Abstract classes	Interfaces
Abstract classes are used only when there is a “is-a” type of relationship between the classes.	Interfaces can be implemented by classes that are not related to one another.
You cannot extend more than one abstract class.	You can implement more than one interface
Abstract class can implement some methods too.	Interfaces can not implement methods [JDK 8: Default methods]

http://www.java-tips.org/java-se-tips/java.lang/difference-between-abstract-classes-and-inter.html

Clicker Question

```
public static void main(String[] args){ m(2); }
private static void m( int counter){
     if(counter > 0){
          System.out.print("hello" + counter);
          m(--counter);
          System.out.print(counter);
      }
}
```

A. hello2 hello1 0 1
B. hello1 hello2 1 0
C. hello1 hello2
D. hello2 hello1

http://danzig.jct.ac.il/java_class/recursion.html

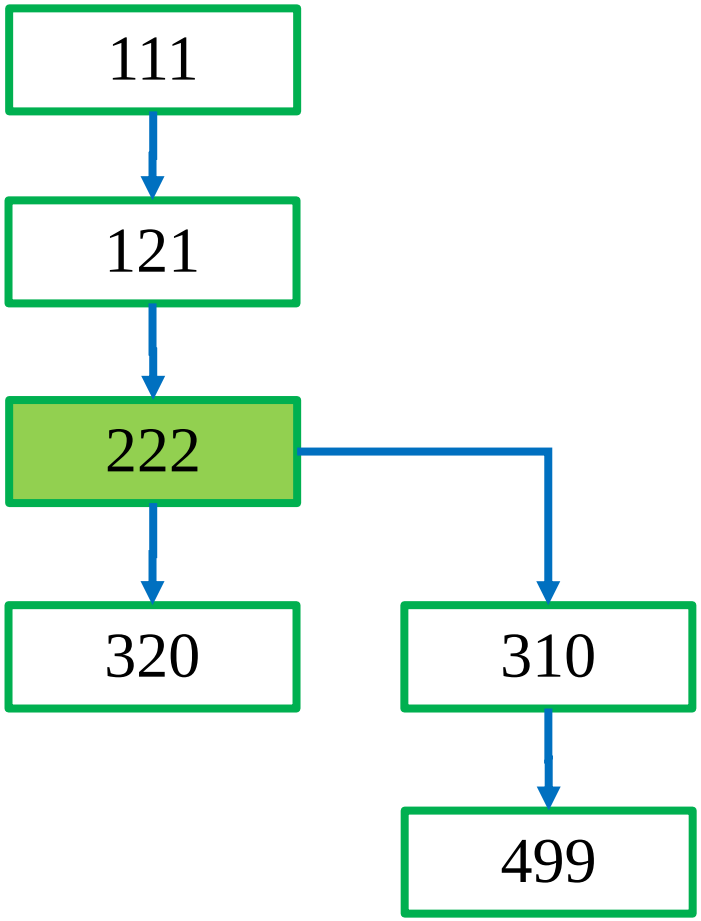
Clicker Question

Fill in the code to complete the following method for computing factorial.

```
public static long factorial(int n) {
  if (n == 0) // Base case
    return 1;
  else
    return _______________; // Recursive call
}
```

A. `n * (n - 1)`
B. `n * factorial(n - 1)`
C. `factorial(n - 1) * n`
D. Either B or C
E. Any of the above

COSC Curriculum



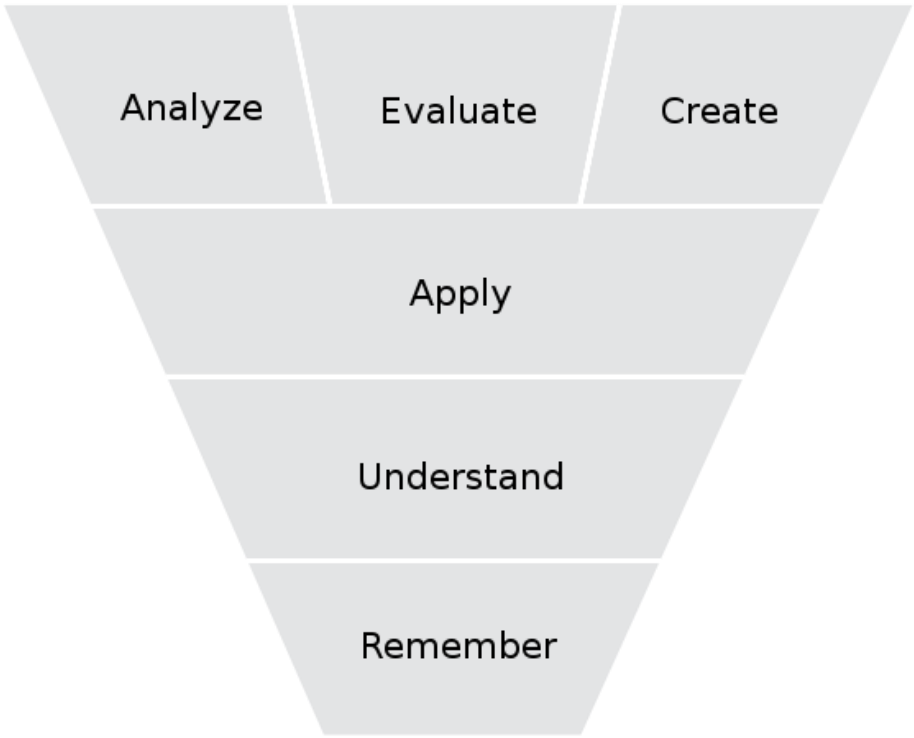
COSC 222 Goals

Solving problems using programming.

Writing efficient programs (in terms of speed and memory usage)

Good programming method:

- problem solving
- design
- testing
- programming



Bloom's Revised Taxonomy

OPPORTUNITIES

Research/Work experience

1) Summer Research Assistant
2) Work-Study program
3) Honours Thesis Course ≥ year 4
4) Directed Studies Course ≥ year 3
5) UBCO URA ≥ 75 credits, …
6) NSERC USRA > Year 1, Prof with NSERC, …
7) Co-op ≥ 70% GPA, …
8) Graduate studies

Opportunities (Paid/Unpaid)

- Research/Project Assistant
- Honours
- Directed Studies

LinkedIn UBCO COSC group

The End

Hope you enjoyed and learned from COSC111/121!