

CP213 Midterm Review Session #2

ARRAYS AND ARRAYLISTS,
INHERITANCE, POLYMORPHISM

Please take this moment to sign in...



UPCOMING EVENTS...

COOP 101 – Nov 2 @ 7:30PM In 1E1

Resume Review – Nov 9 @ 7:30PM On Twitch

Next Review Session:

Nov 15 @ 7:30PM On Twitch

Useful Links and Follow Along Resources

MATH AND STATS LEARNING SUPPORT

This office hosts drop in and directed homework sessions to give you any extra help you need with course content in most lower level CS courses. You can also get their course widget in MyLS.

CODING SANDBOXES

A really good Java code sandbox is Replit.com. Follow along or write code on your own time without setting up files by using one of these.

A really good **Python, Java, HTML**, sandbox (and almost every other language) is Replit.com

<https://replit.com/>

Another good **HTML/CSS/JS** sandbox:

<https://codepen.io/>

Mental Health Resources

We know university is very challenging and difficult. Please don't hesitate to reach out to people if you need help. Listed below are a few useful resources you can access, for additional resources please view the LCS linktree.

SAFEHAWK APP

Connects you with telephone helplines and other mental health resources.

STUDENT WELLNESS CENTRE

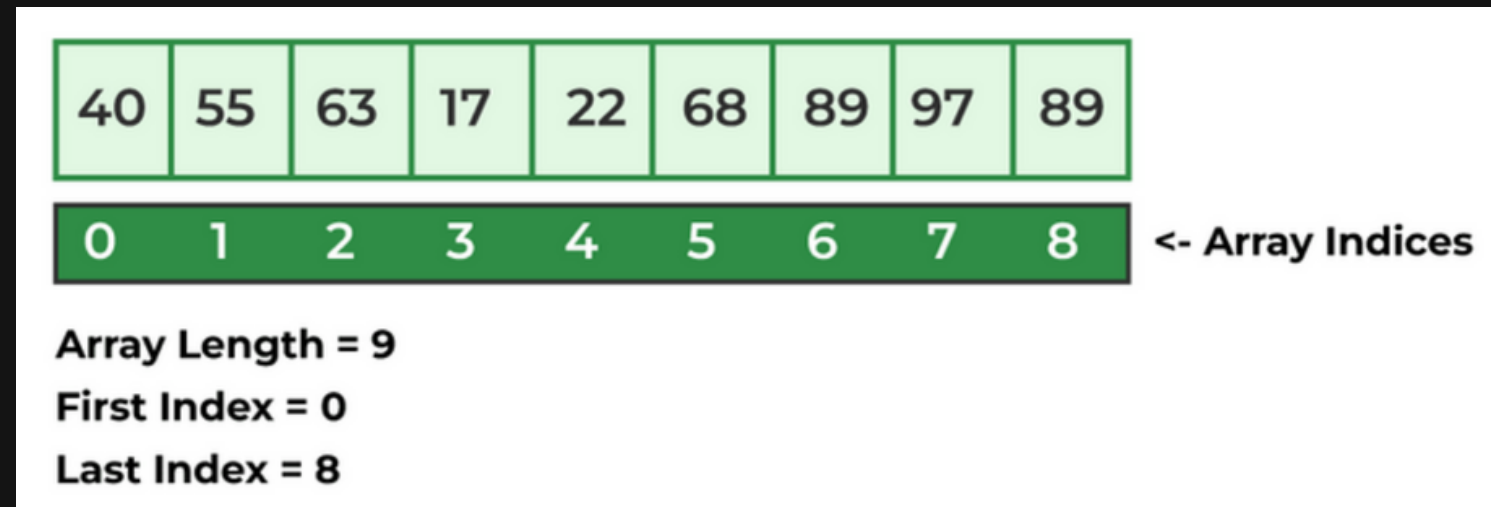
Provides comprehensive physical, emotional and mental health services for Waterloo and Brantford Campus students.

DELTON GLEBE COUNSELLING CENTRE:

A holistic counselling facility.

Arrays: Introduction

- AN ARRAY IS A DATA STRUCTURE USED TO PROCESS A COLLECTION OF DATA THAT IS ALL OF THE SAME TYPE






- AN ARRAY THAT BEHAVES LIKE THIS COLLECTION OF VARIABLES, ALL OF TYPE **DOUBLE**, CAN BE CREATED USING ONE STATEMENT AS FOLLOWS:
- **DOUBLE[] SCORE = NEW DOUBLE[5];**
- OR USING TWO STATEMENTS:
 - **DOUBLE[] SCORE;**
 - **SCORE = NEW DOUBLE[5];**

Arrays Cont'd

- THE INDIVIDUAL VARIABLES THAT TOGETHER MAKE UP THE ARRAY ARE CALLED INDEXED VARIABLES
- IN JAVA, INDICES MUST BE NUMBERED STARTING WITH 0, AND NOTHING ELSE
 - EX: SCORE[0], SCORE[1], SCORE[2], SCORE[3], SCORE[4]
- JAVA ARRAYS ARE OBJECTS!!!
 - NOT ALL LANGUAGES HAVE ARRAYS AS OBJECTS
- AN ARRAY HAS ITS LENGTH FIXED WHEN INITIALISED
- LIKE PYTHON, JAVA ARRAYS CAN BE MULTIDIMENSIONAL

Arrays: Example

long form

```
double[] a;  declaration  
a = new double[N];  creation  
for (int i = 0; i < N; i++)  
    a[i] = 0.0;  initialization
```

short form

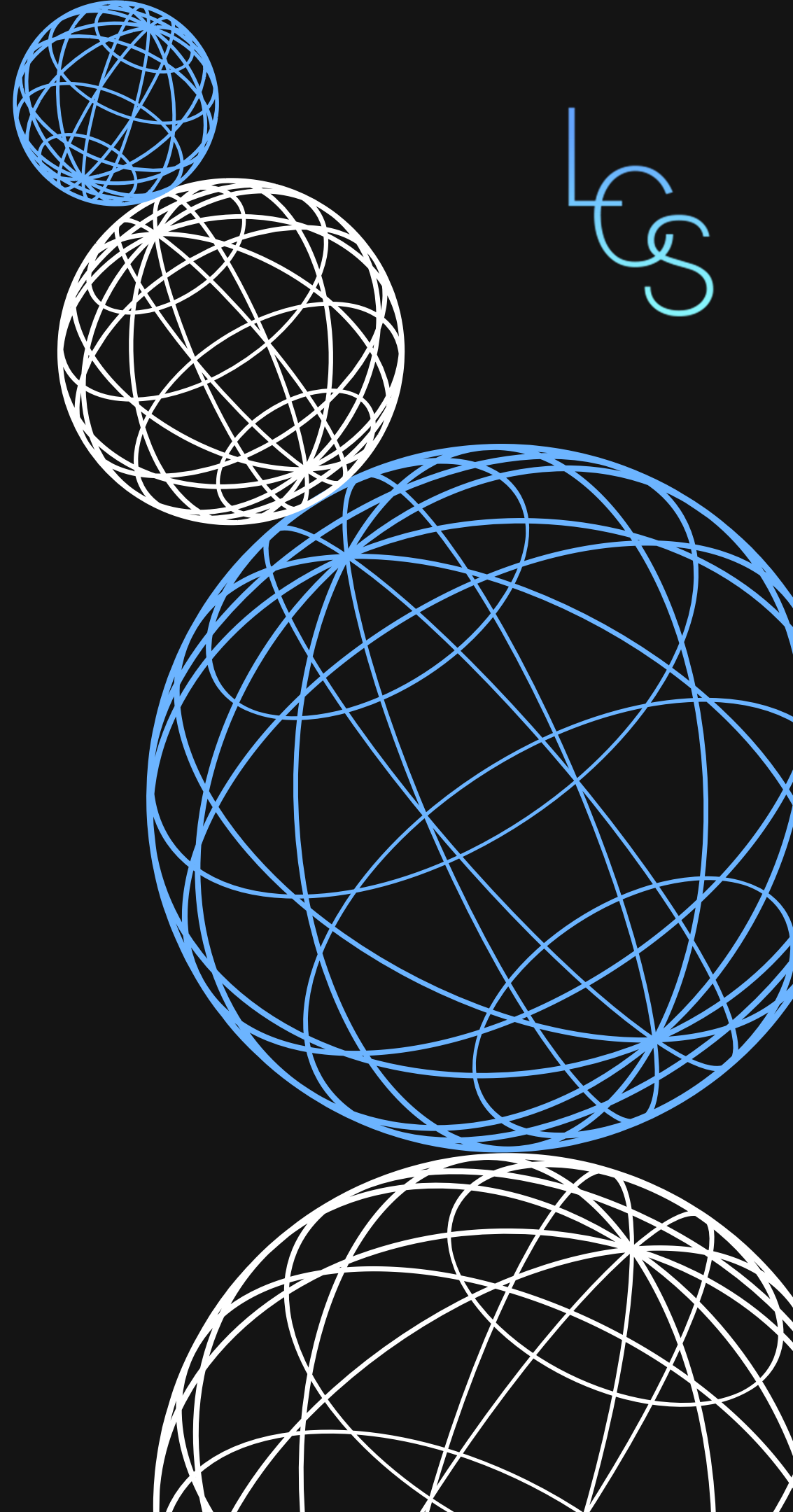
```
double[] a = new double[N];
```

initializing declaration

```
int[] a = { 1, 1, 2, 3, 5, 8 };
```

- WHEN YOU CREATE AN ARRAY IN JAVA, YOU'RE ESSENTIALLY CREATING AN OBJECT OF THE CORRESPONDING ARRAY CLASS, AND YOU CAN USE METHODS AND PROPERTIES PROVIDED BY THAT CLASS TO WORK WITH THE ARRAY

**Do you have any
questions?**



ArrayList

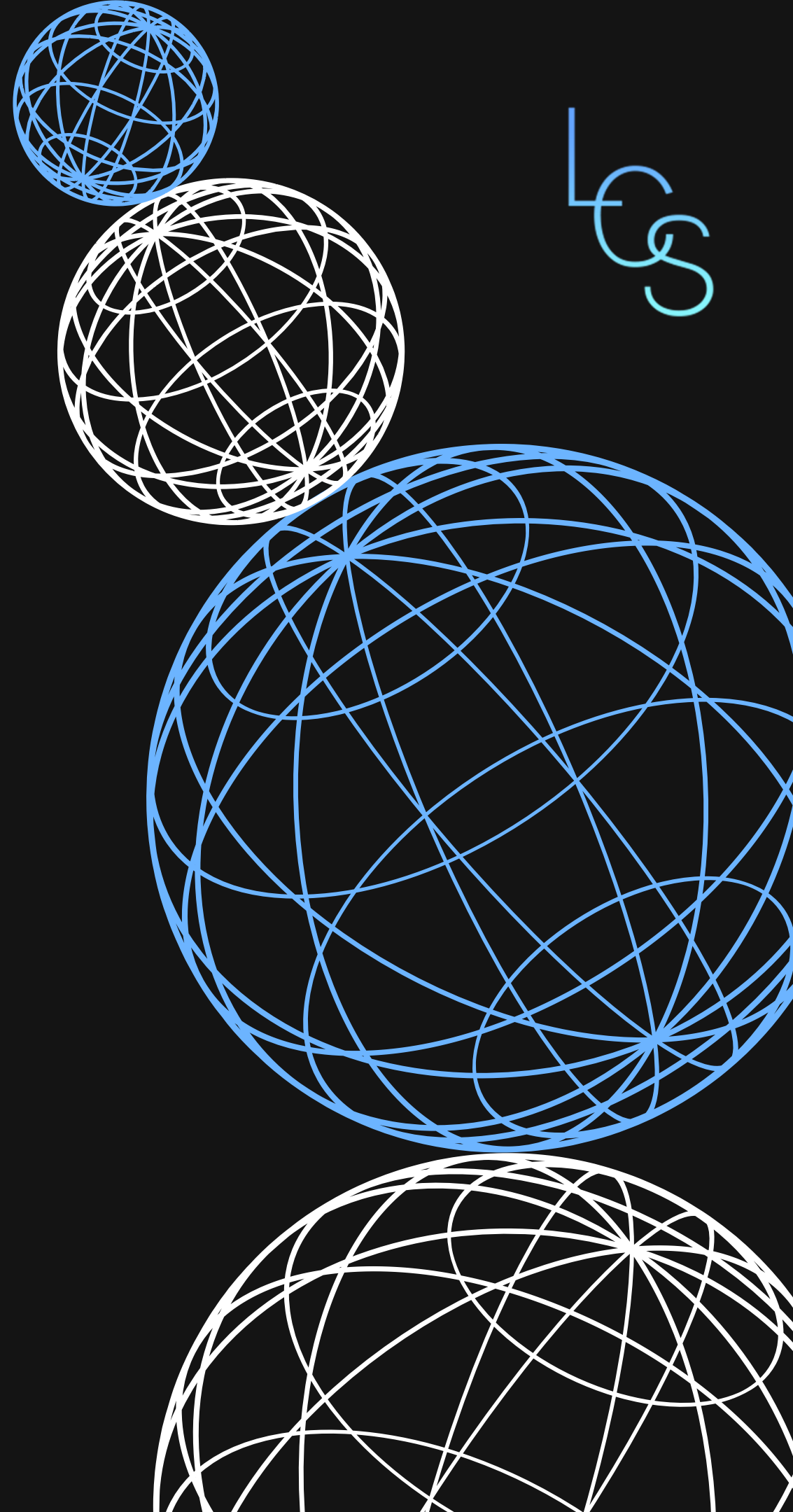
- **ARRAYLIST:** A JAVA DATA STRUCTURE AND OBJECT WHICH ACTS LIKE AN ARRAY WITH A DYNAMIC (CAN BE CHANGED) LENGTH
- **ARRAYLIST<STRING>** REPRESENTS ARRAYLIST OF STRINGS.
- **ARRAYLIST<TYPE> LIST = NEW ARRAYLIST<TYPE>();**
 - INITIALISE
- METHODS DEMONSTRATED ON THE NEXT SLIDE

ArrayList Methods

- `LIST.ADD(ELEMENT);`
 - THIS LETS YOU ADD THE ELEMENT TO THE ARRAYLIST
- `TYPE ELEMENT = LIST.GET(INDEX);`
 - YOU CAN ACCESS ELEMENTS IN THE ARRAYLIST BY THEIR INDEX USING THE GET METHOD
- `INT SIZE = LIST.SIZE();`
 - RETURNS SIZE OF ARRAYLIST
- REMOVE METHOD
 - `LIST.REMOVE(INDEX);` TO REMOVE BY INDEX
 - `LIST.REMOVE(ELEMENT);` TO REMOVE AN ELEMENT

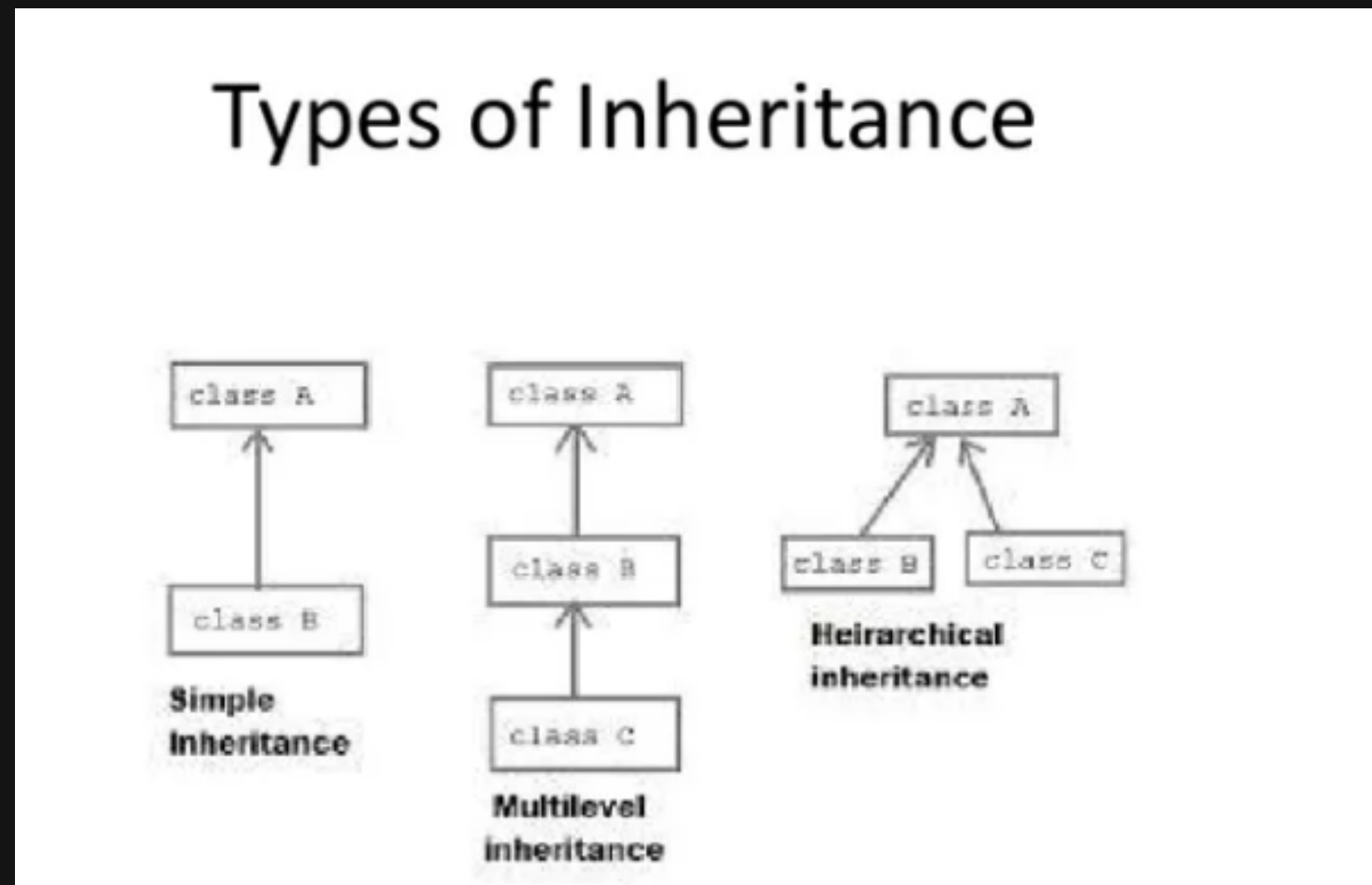
Arrays vs ArrayLists: Summary

**Do you have any
questions?**



Inheritance

- **INHERITANCE** IS ONE OF THE MAIN TECHNIQUES OF OBJECT-ORIENTED PROGRAMMING (OOP)
- USING THIS TECHNIQUE, A VERY GENERAL FORM OF A CLASS IS FIRST DEFINED AND COMPILED, AND THEN MORE SPECIALIZED VERSIONS OF THE CLASS ARE DEFINED BY ADDING INSTANCE VARIABLES AND METHODS
 - THE SPECIALIZED CLASSES ARE SAID TO INHERIT THE METHODS AND INSTANCE VARIABLES OF THE GENERAL CLASS



Syntax

```
// Java Program to illustrate Inheritance (concise)

import java.io.*;

// Base or Super Class
class Employee {
    int salary = 60000;
}

// Inherited or Sub Class
class Engineer extends Employee {
    int benefits = 10000;
}
```

- BY USING THE 'EXTEND' KEYWORD WE ARE EXTENDING THE ORIGINAL CLASS
- IN THE EXAMPLE ABOVE, WE HAVE A CLASS EMPLOYEE AND WE ARE EXTENDING THAT CLASS TO ENGINEER
- AN ENGINEER IS A TYPE OF EMPLOYEE THERE IS USES THAT CLASS'S GENERIC ATTRIBUTES BUT IT ALSO REQUIRES ITS OWN SET OF ATTRIBUTES

Example

```
// Java program to illustrate the
// concept of inheritance

// base class
class Bicycle {
    // the Bicycle class has two fields
    public int gear;
    public int speed;

    // the Bicycle class has one constructor
    public Bicycle(int gear, int speed)
    {
        this.gear = gear;
        this.speed = speed;
    }

    // the Bicycle class has three methods
    public void applyBrake(int decrement)
    {
        speed -= decrement;
    }

    public void speedUp(int increment)
    {
        speed += increment;
    }

    // toString() method to print info of Bicycle
    public String toString()
    {
        return ("No of gears are " + gear + "\n"
            + "speed of bicycle is " + speed);
    }
}
```

```
// derived class
class MountainBike extends Bicycle {

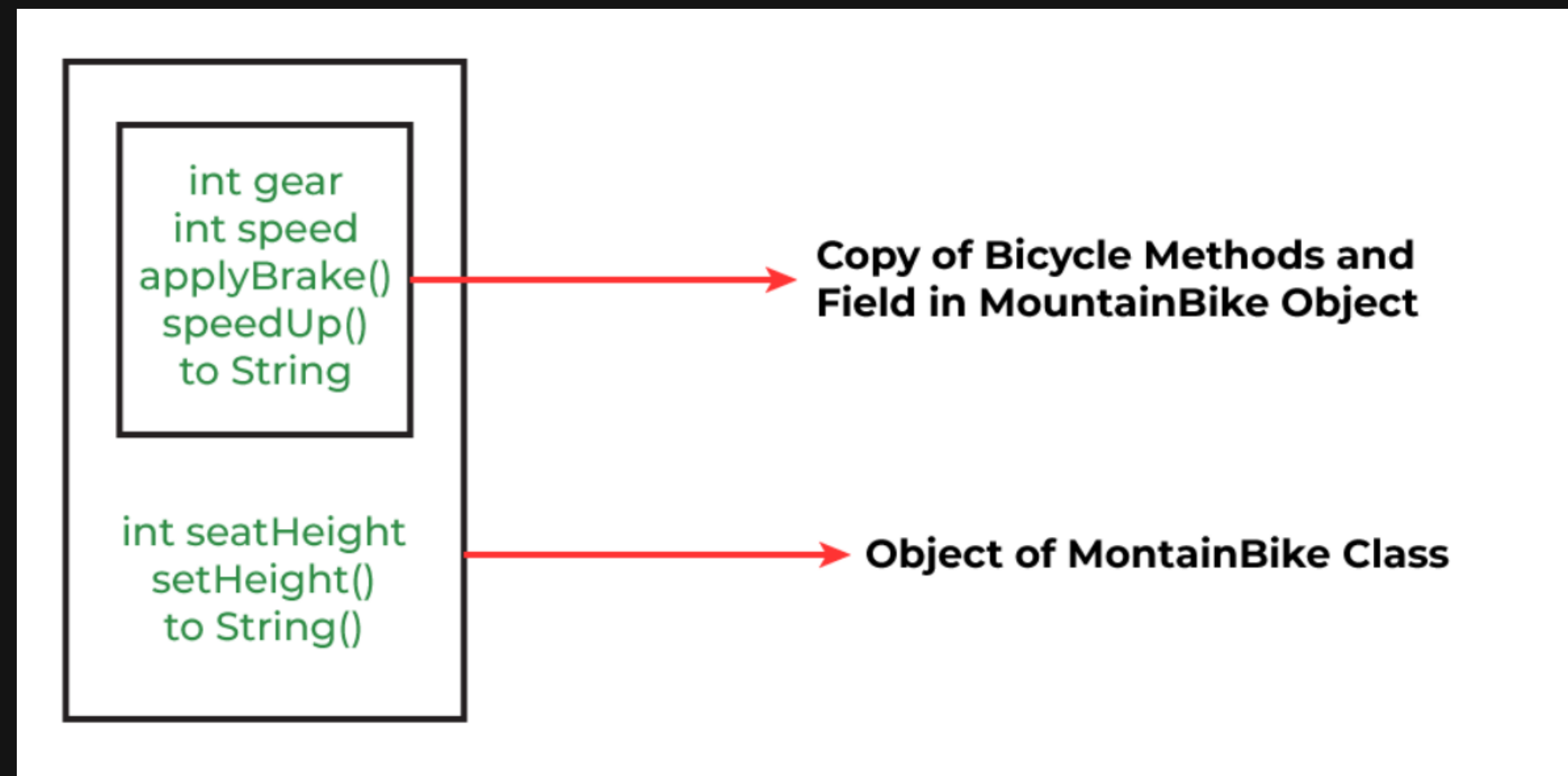
    // the MountainBike subclass adds one more field
    public int seatHeight;

    // the MountainBike subclass has one constructor
    public MountainBike(int gear, int speed,
        int startHeight)
    {
        // invoking base-class(Bicycle) constructor
        super(gear, speed);
        seatHeight = startHeight;
    }

    // the MountainBike subclass adds one more method
    public void setHeight(int newValue)
    {
        seatHeight = newValue;
    }

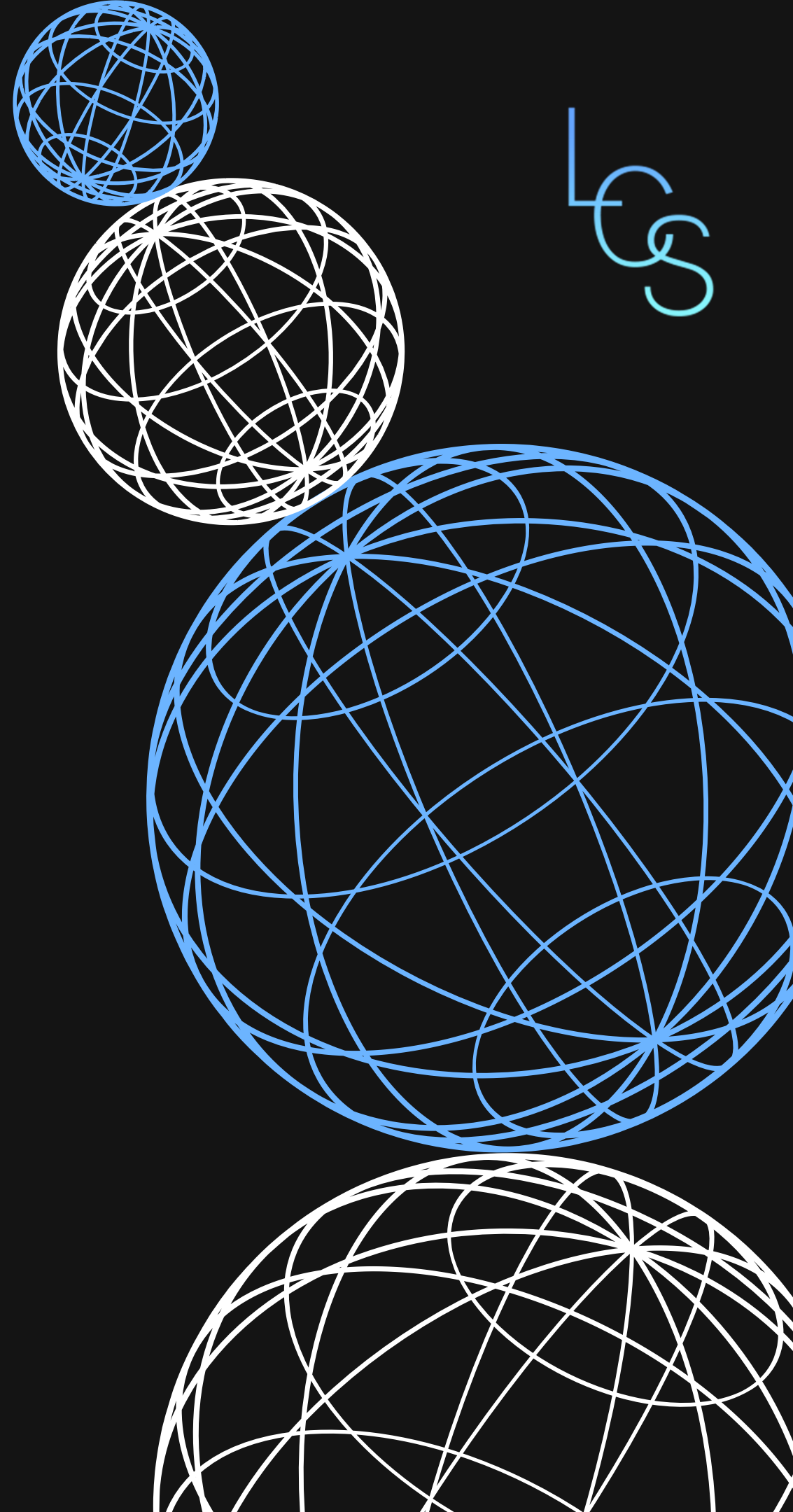
    // overriding toString() method
    // of Bicycle to print more info
    @Override public String toString()
    {
        return (super.toString() + "\nseat height is "
            + seatHeight);
    }
}
```

- CLASS BICYCLE IS A BASE CLASS, CLASS MOUNTAINBIKE IS A DERIVED CLASS THAT EXTENDS THE BICYCLE CLASS



- IN THE ABOVE PROGRAM, WHEN AN OBJECT OF MOUNTAINBIKE CLASS IS CREATED, A COPY OF ALL METHODS AND FIELDS OF THE SUPERCLASS ACQUIRES MEMORY IN THIS OBJECT. THAT IS WHY BY USING THE OBJECT OF THE SUBCLASS WE CAN ALSO ACCESS THE MEMBERS OF A SUPERCLASS.

**Do you have any
questions?**



Polymorphism

- INHERITANCE ALLOWS A BASE CLASS TO BE DEFINED, AND OTHER CLASSES DERIVED FROM IT
 - CODE FOR THE BASE CLASS CAN THEN BE USED FOR ITS OWN OBJECTS, AS WELL AS OBJECTS OF ANY DERIVED CLASSES
- POLYMORPHISM ALLOWS CHANGES TO BE MADE TO METHOD DEFINITIONS IN THE DERIVED CLASSES, AND HAVE THOSE CHANGES APPLY TO THE SOFTWARE WRITTEN INITIALLY FOR THE BASE CLASS
- BELOW IS AN EXAMPLE

The **Sale** and **DiscountSale** Classes

- The **Sale** class **bill()** method:

```
public double bill( )  
{  
    return price;  
}
```

- The **DiscountSale** class **bill()** method:

```
public double bill( )  
{  
    double fraction = discount/100;  
    return (1 - fraction) * getPrice( );  
}
```

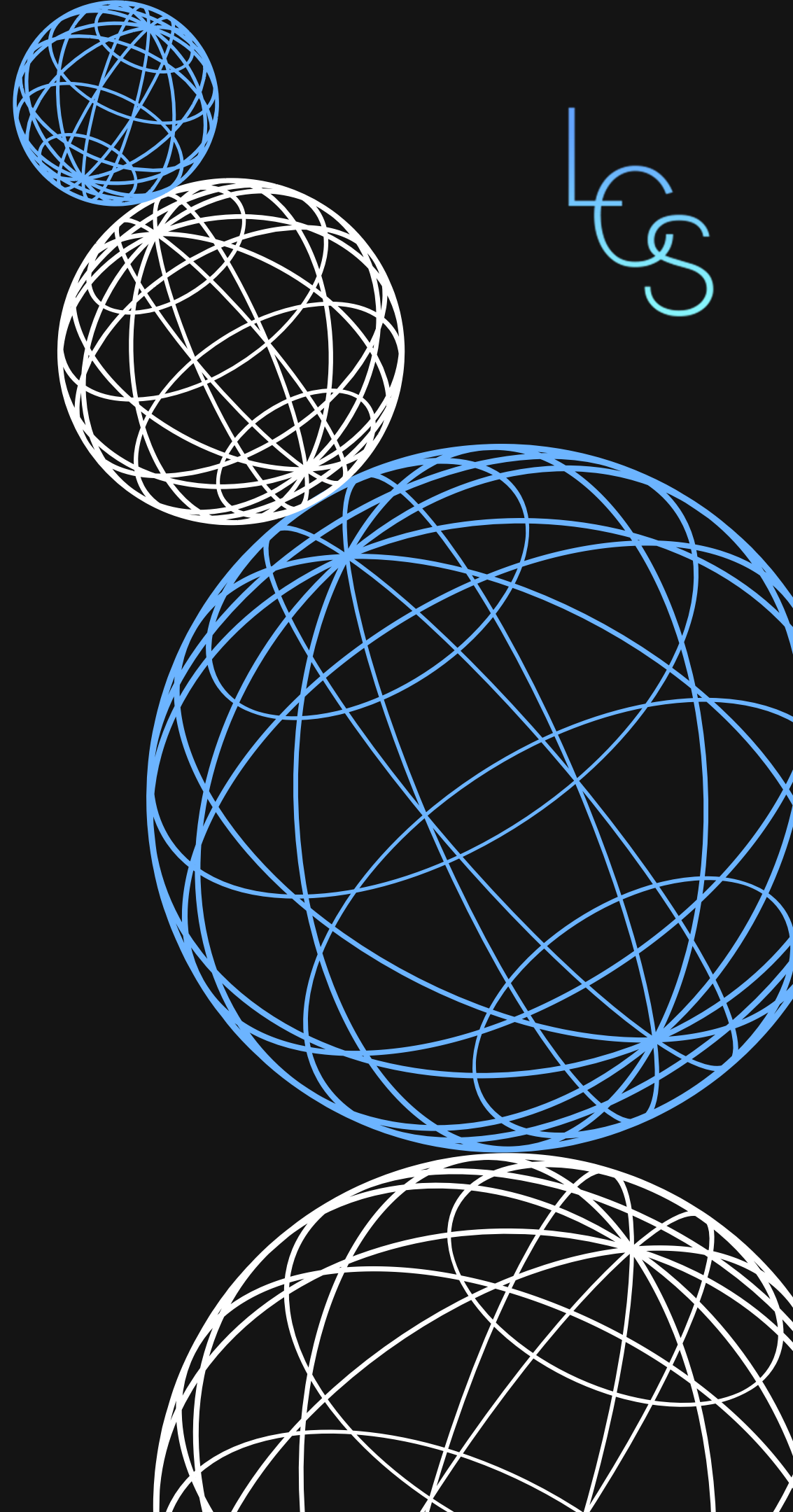
This is polymorphism, as the method has multiple (“poly”) forms (“morphisms”) between the child and parent classes

Binding

- THE PROCESS OF ASSOCIATING METHOD DEFINITION WITH A METHOD CALL IS CALLED BINDING
- IF BINDING IS DONE AT COMPILE TIME, THE PROCESS IS CALLED EARLY BINDING, OR STATIC BINDING
- IF BINDING IS DONE AT RUN TIME, THE PROCESS IS CALLED LATE BINDING, OR DYNAMIC BINDING
 - COMPILE TIME IS THE PERIOD WHEN THE HIGH-LEVEL CODE IS CONVERTED TO THE MACHINE CODE
 - RUNTIME IS THE PERIOD OF TIME WHEN A PROGRAM IS RUNNING AND OCCURS AFTER COMPILE TIME

```
public class Student {  
  
    public void registerCourse (int courseNumber) {  
        // code goes here  
    }  
  
    public static void main (String args []) {  
        Student aStudent = new Student ();  
        aStudent.registerCourse(1020);  
    }  
}
```

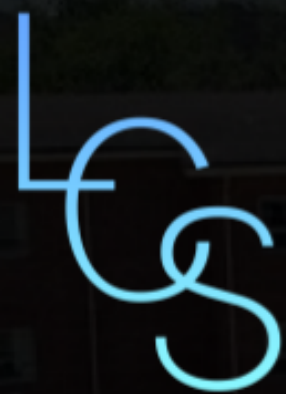
**Do you have any
questions?**



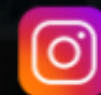
Please fill out this event survey!



Thank you for coming!



Laurier
Computing
Society



@laurier.cs



discord.lauriercs.ca



linktr.ee/lauriercs

