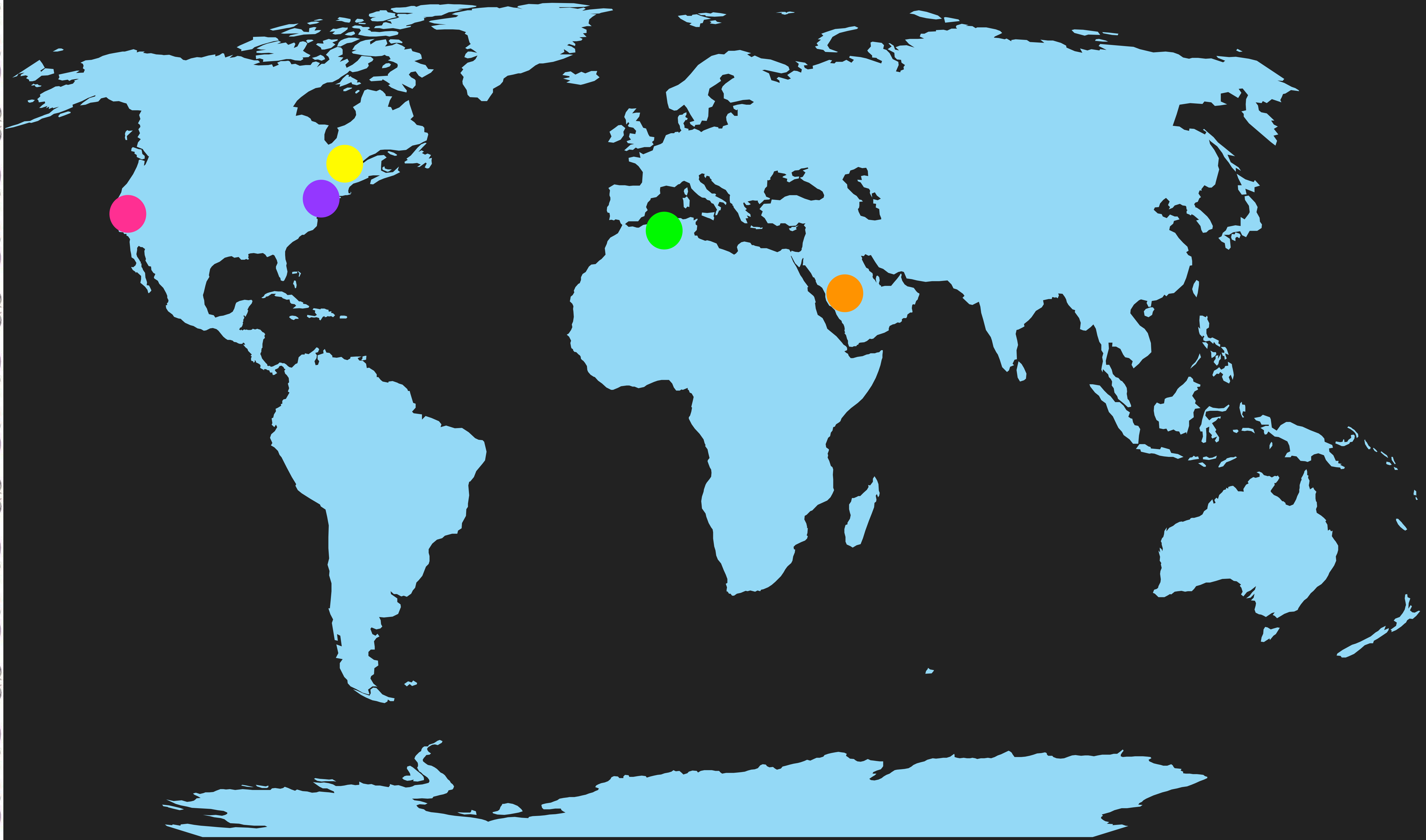
PROGRAMMING AND DATA STRUCTURES

INTRODUCTION

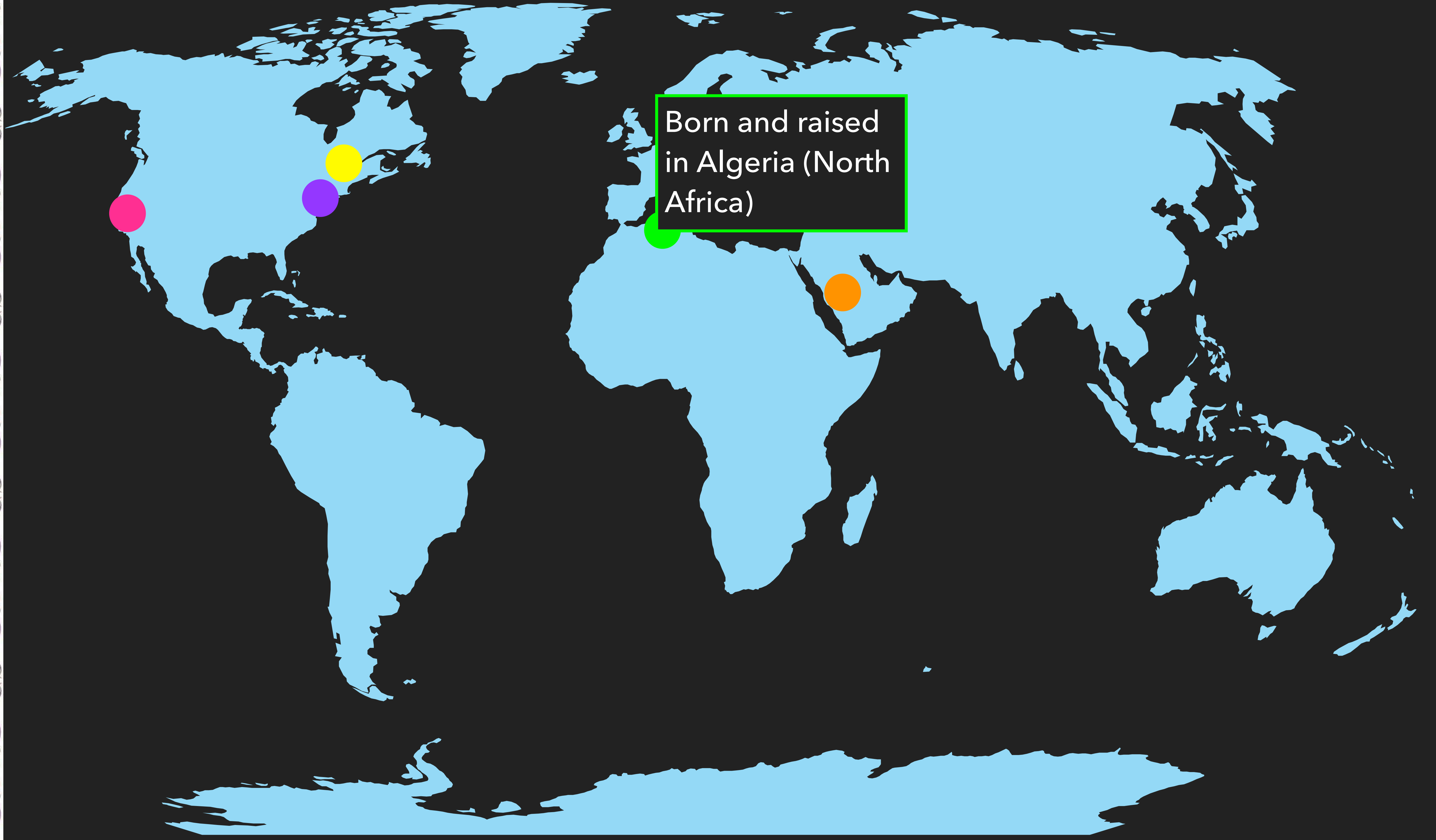
HOURIA OUDGHIRI

FALL 2021

WHO AM I?

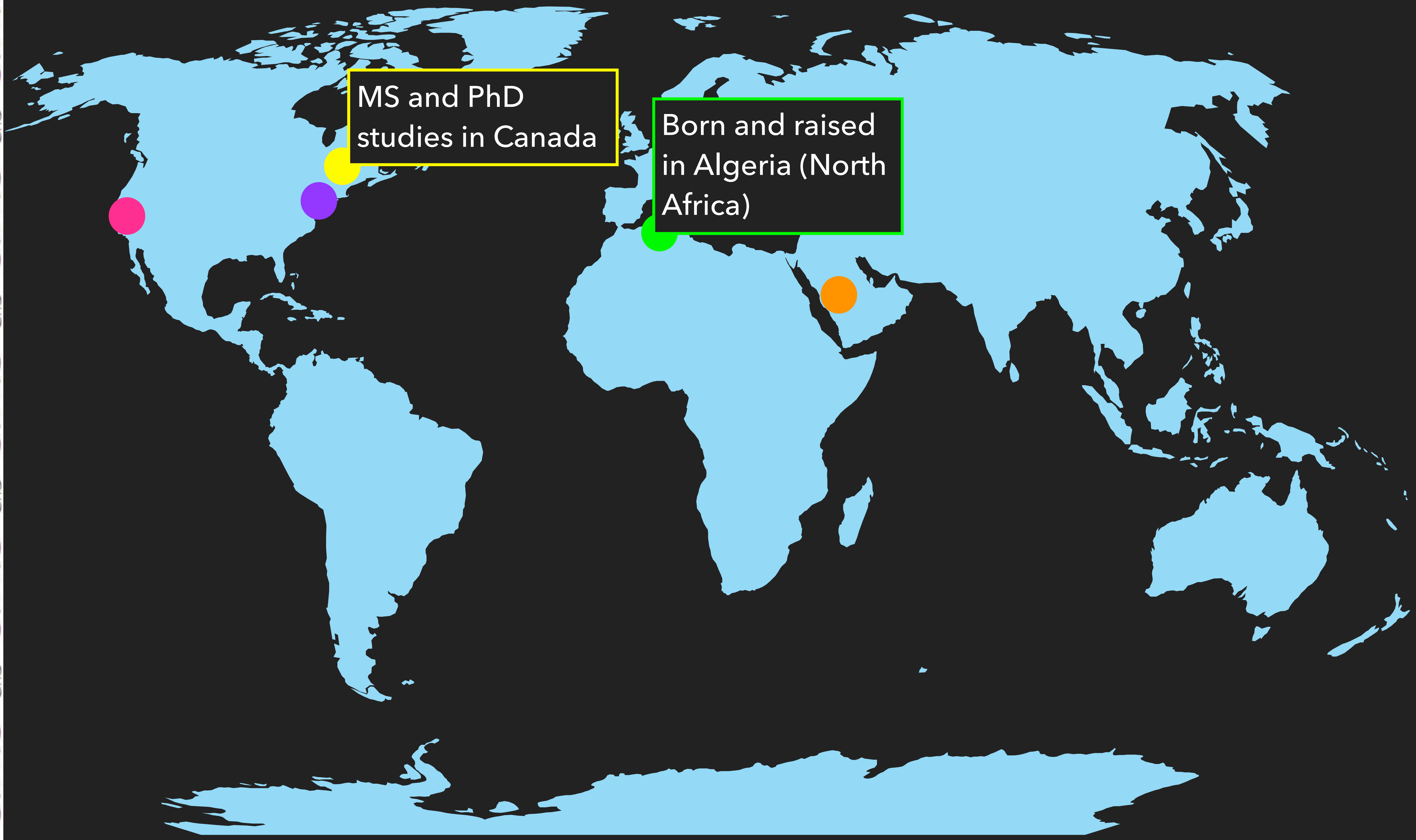


WHO AM I?

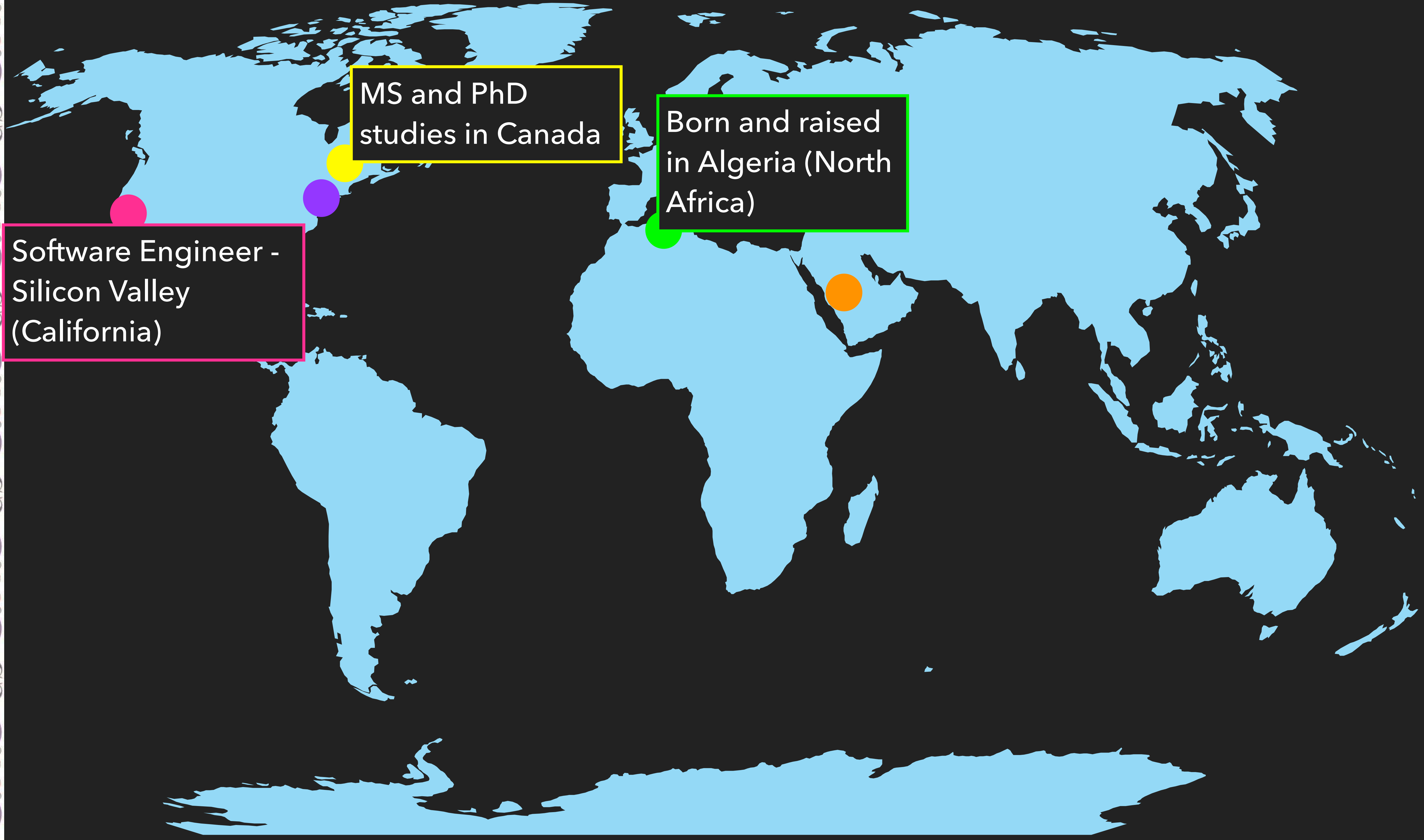


Born and raised in Algeria (North Africa)

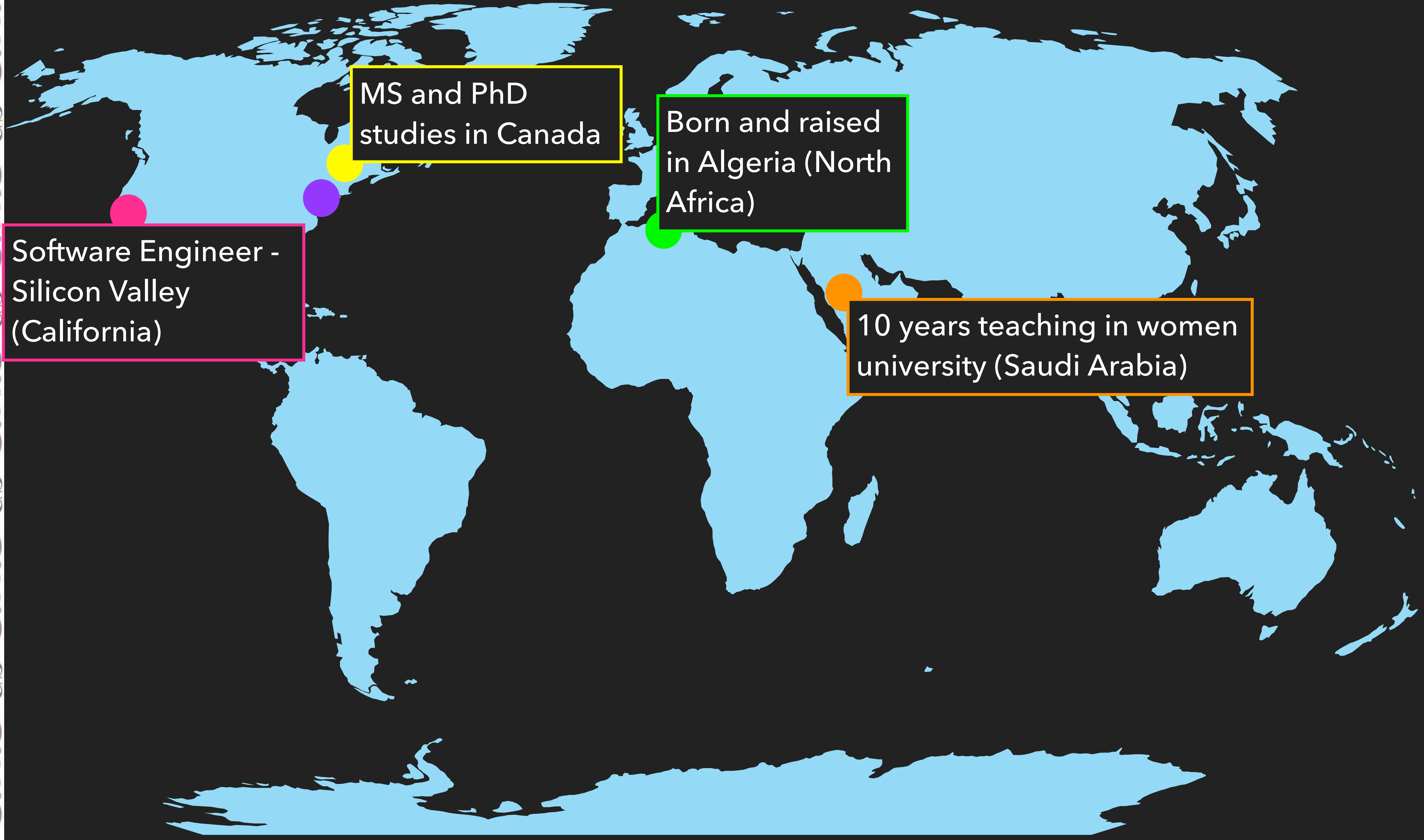
WHO AM I?


MS and PhD studies in Canada
Born and raised in Algeria (North Africa)

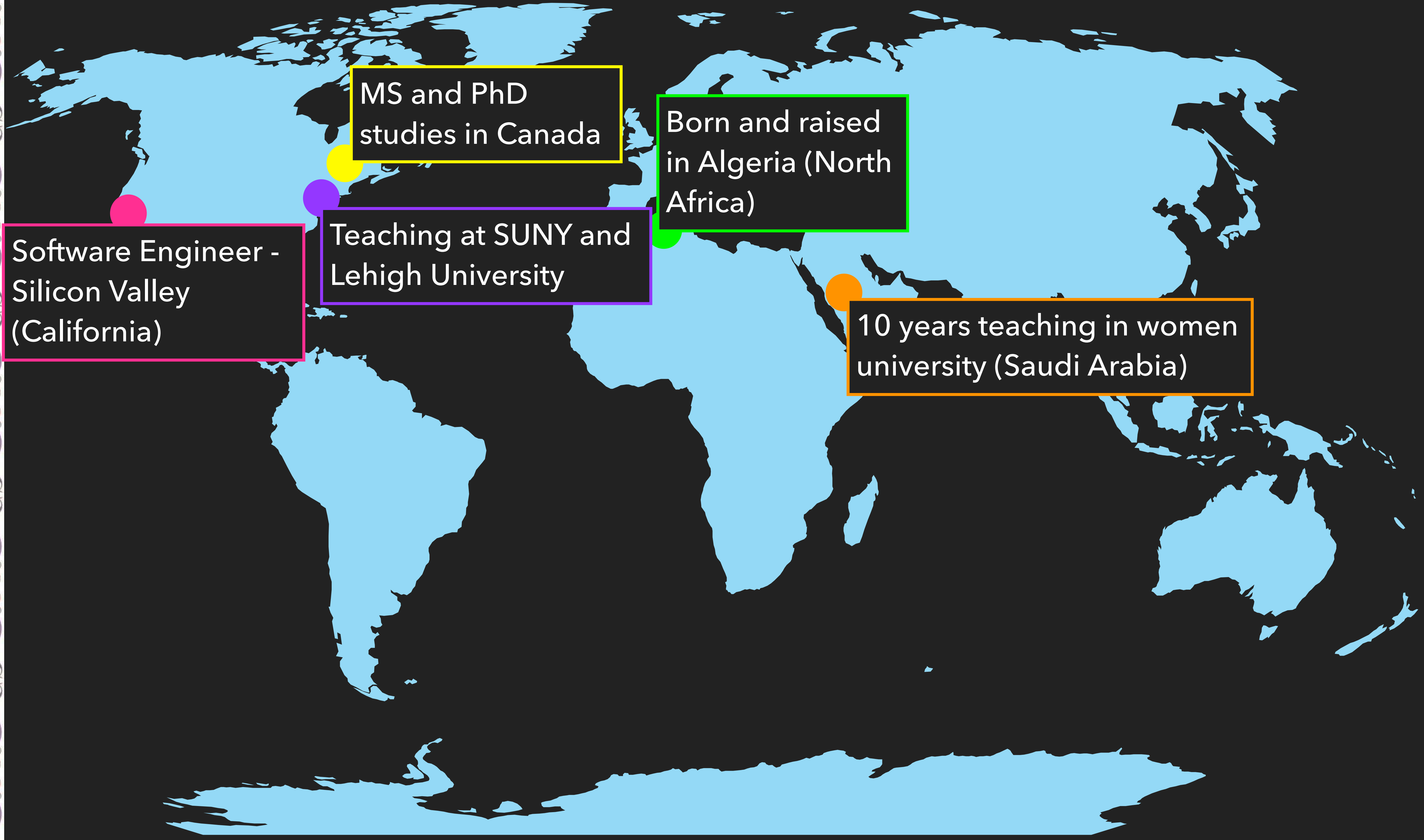
WHO AM I?


MS and PhD studies in Canada
Born and raised in Algeria (North Africa)
Software Engineer - Silicon Valley (California)

WHO AM I?


MS and PhD studies in Canada
Born and raised in Algeria (North Africa)
Software Engineer - Silicon Valley (California)
10 years teaching in women university (Saudi Arabia)

WHO AM I?


MS and PhD studies in Canada
Born and raised in Algeria (North Africa)
Software Engineer - Silicon Valley (California)
Teaching at SUNY and Lehigh University
10 years teaching in women university (Saudi Arabia)

OUTLINE

- What is CSE017?
- Student Learning Outcomes
- Course syllabus
- Review of Java and OOP Fundamentals

WHAT IS CSE017?

- Programming and Data Structures
- CSE3/4/7 Programming Fundamentals
  - One class with a main method and sometimes more methods
  - Creating/Instantiating/Extending classes

WHAT IS CSE017?

- Useful classes in Java - OOP applications (Exception handling and File I/O)
- Special classes - Abstract classes/Interfaces
- Classes to store and manipulate data - Data Structures
- Algorithms to manipulate data - Recursion, Searching and Sorting

STUDENT LEARNING OUTCOMES

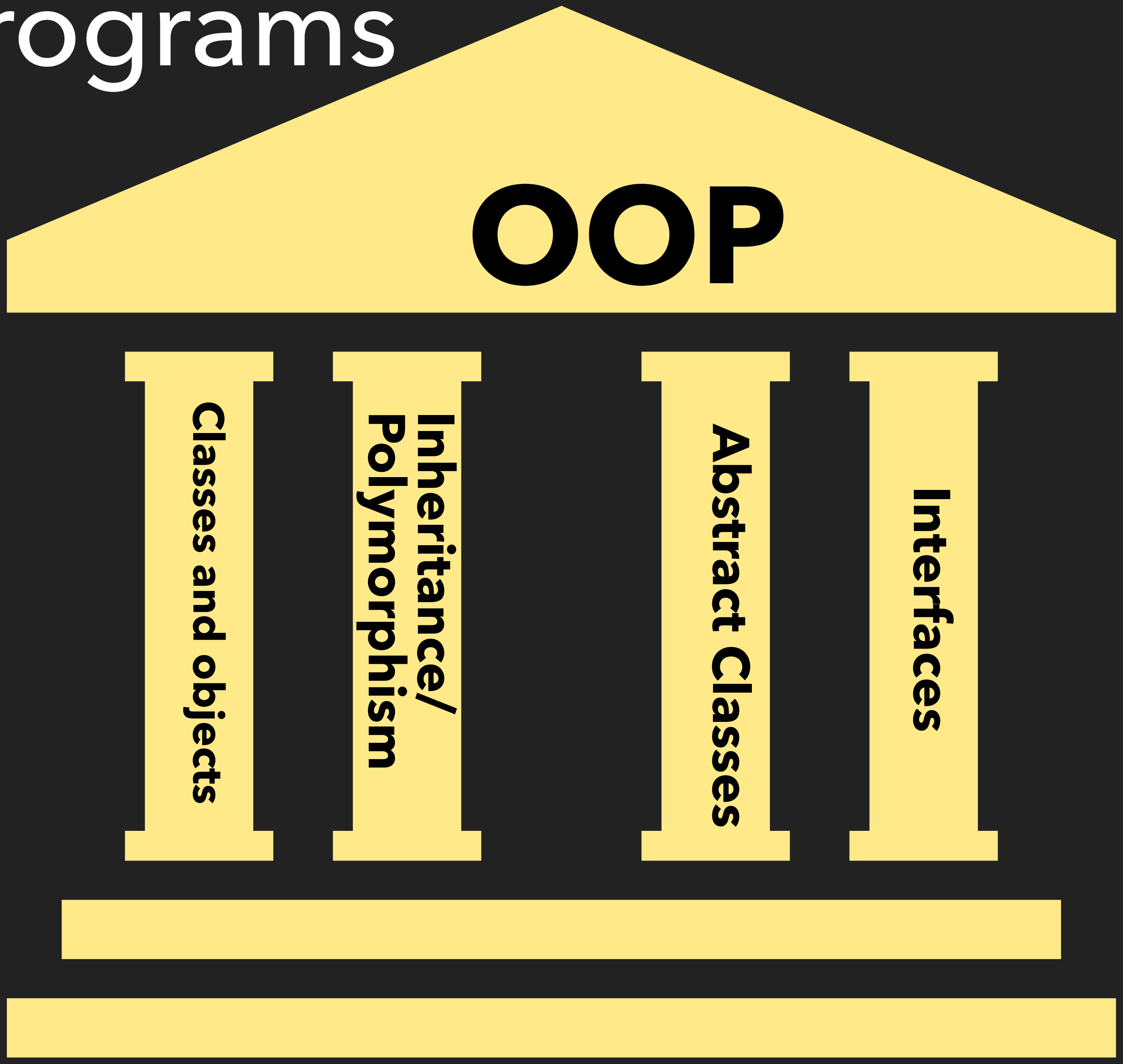
- What knowledge and skills would you acquire by the end of the course?

STUDENT LEARNING OUTCOMES

1. Apply object oriented programming to design Java programs
2. Design and implement data structures for data storage and manipulation
3. Use recursion to implement algorithms
4. Implement sorting algorithms and compare them using algorithm analysis techniques

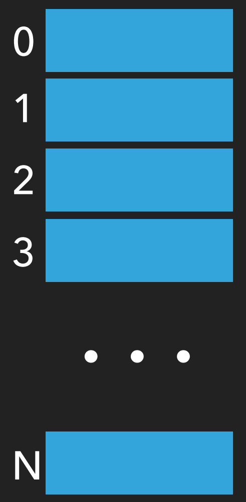
STUDENT LEARNING OUTCOMES

1. Apply Object Oriented Concepts to write Java programs

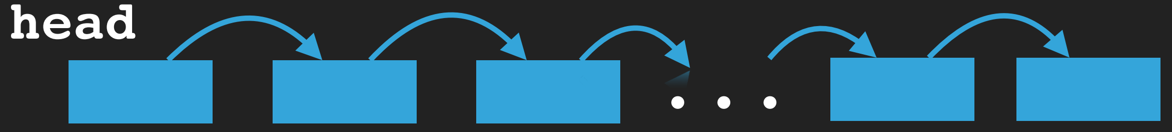


STUDENT LEARNING OUTCOMES

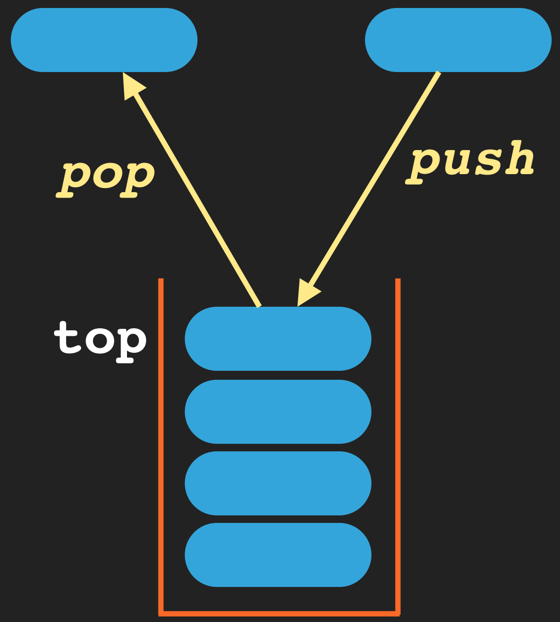
2. Implement common data structures to store and manipulate data



Array List



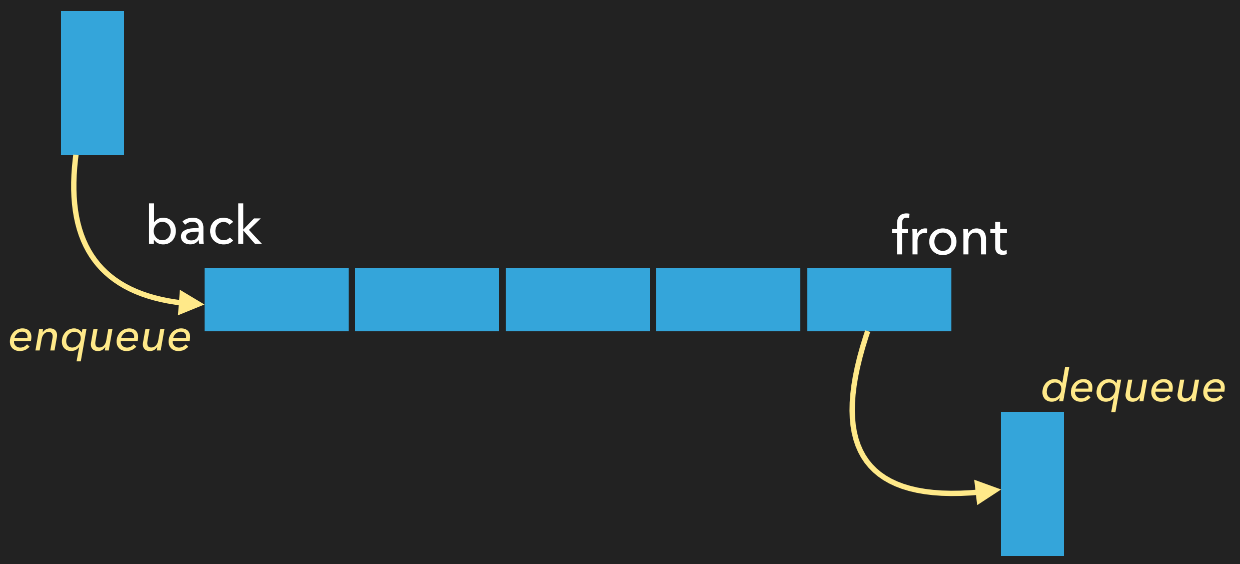
Linked List



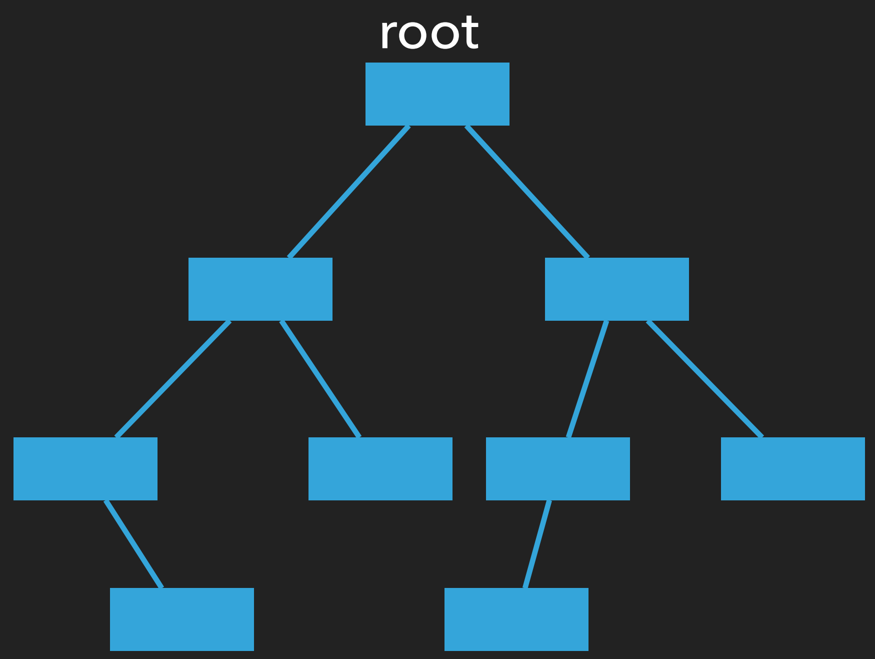
Stack

STUDENT LEARNING OUTCOMES

2. Implement common data structures to store and manipulate data



Queue

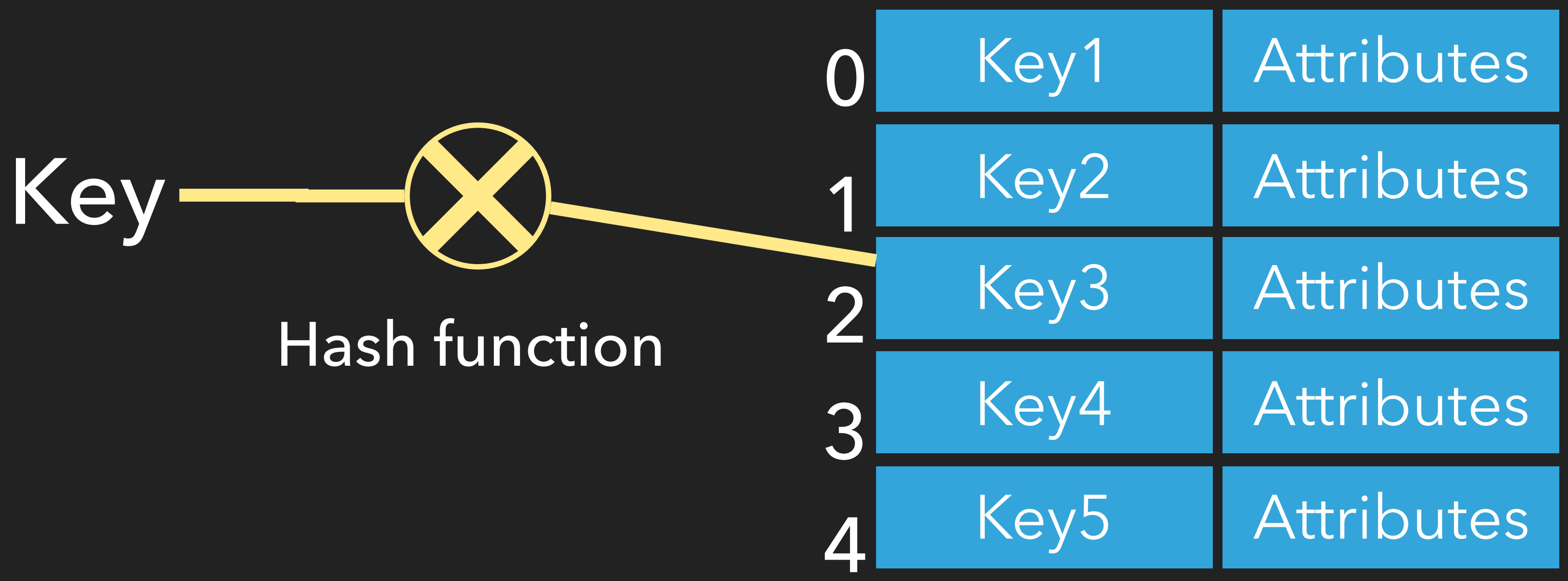


Tree

Tree

STUDENT LEARNING OUTCOMES

2. Implement common data structures to store and manipulate data



HashTable (HashMap)

Tree

STUDENT LEARNING OUTCOMES

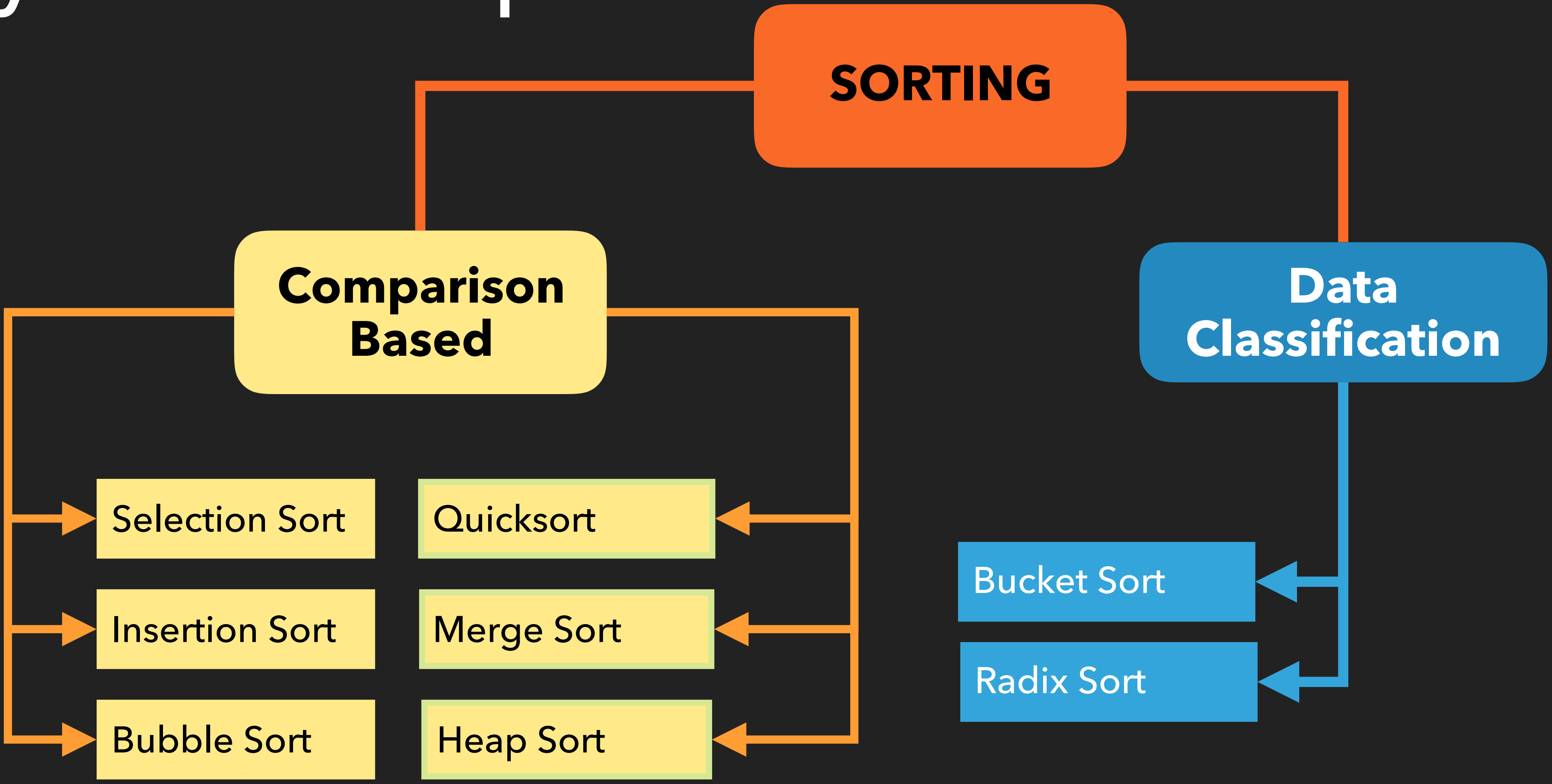
3. Use recursion to implement algorithms

```
public class Test {
 public static void main(String[] args){
  int n = 10;
  System.out.println("Sum: " + sum(n));
 }
 public static int sum(int n){
  int s = 0;
  for(int i=1; i<= n; i++)
    s += i;
  return s;
 }
}
```

```
public class Test {
 public static void main(String[] args){
  int n = 10;
  System.out.println("Sum: " + sum(n));
 }
 public static int sum(int n){
  if (n == 1)
    return 1;
   else
    return n + sum(n-1);
  }
}
```

STUDENT LEARNING OUTCOMES

4. Implement and compare different sorting algorithms using algorithm analysis techniques



COURSE SYLLABUS

JAVA/OOP OVERVIEW

- Types/Operators/Assignment Statements
- Selection/Iteration Control Statements
- Input and Output
- Methods
- Arrays
- Object Oriented Programming (Classes and Objects, Inheritance, Polymorphism)

Data Types and Operators

- Data types -
  - Primitive types - `int`, `float`, `double`, `char`, `byte`, `boolean`
  - Class type - `String, Scanner`
- Arithmetic - `+, -, *, /, %, ++, --`
- Logical - `&&, ||, !`
- Relational - `<, <=, >, >=, ==, !=`
- Conditional operator - `? :`

Assignment Statements

- Simple - `= (x = 20)`
- Compound - `+=, -=, *=, /=, %=`

  `x *= 10 (x = x * 10)`

JAVA OVERVIEW

- What is the output of the following Java code for **x = 2**, **y = 3** and **z = 6**?

```
public class Test {
   public static void main(String[] args) {
      double x, y, z;
      java.util.Scanner input = new java.util.Scanner(System.in);
      x = input.nextDouble();
      y = input.nextDouble();
      z = input.nextDouble();
      System.out.println("(x > y && y < z) is " + (x > y && y < z));
      System.out.println("(x > y || y < z) is " + (x > y || y < z));
      System.out.println("!(x >= y) is " + !(x >= y));
      System.out.println("(2x + y/2 < z) is " + (2 * x + y / 2) < z));
      input.close();
   }
}
```

JAVA OVERVIEW

- What is the value of each of the following expressions?

```
2 * 2 - 3 < 2 && 4 - 2 > 5


3 + 4 * 4 > 5 * (4 + 3) - 1 && 9 - 2 > 5
```

JAVA OVERVIEW

- What is the output of the following Java code for **x = 8** and **number = 12**?

```
public class Test {
   public static void main(String[] args) {
      int x, score, scale=10;
      java.util.Scanner input = new java.util.Scanner(System.in);
      x = input.nextInt();
      score = (x > 10) ? 3 * scale : 4 * scale;
      System.out.println("score = " + score);
      x = input.nextInt();
      System.out.println((x % 3 == 0) ? 27 : 25);
      input.close();
   }
}
```

Selection/Iteration Control Statements

- If - else statement - one/two alternatives
- Nested Ifs - multiple alternatives
- Switch statement - multiple alternatives for integer and character/string type expressions
- Loops - for/while/do-while
- Nested loops
- Break/Continue statements

JAVA OVERVIEW

- What is the output of the following Java code for **score = 80**?

```
public class Test {
   public static void main(String[] args) {
      java.util.Scanner input = new java.util.Scanner(System.in);
      double score = input.nextDouble();
      if (score >= 60)
        System.out.println("D");
      else if (score >= 70)
        System.out.println("C");
      else if (score >= 80)
        System.out.println("B");
      else if (score >= 90)
        System.out.println(“A”);
      else
        System.out.println("F");
      input.close();
   }
}
```

JAVA OVERVIEW

- What is the output of the following Java code?

```
public class Test{
  public static void main(String[] args){
    for (int i = 1; i < 5; i++){
      int j = 0;
      while (j < i){
        System.out.print(j + " ");
        j++;
      }
      System.out.println();
    }
  }
}
```

JAVA OVERVIEW

- What is the output of the following Java code?

```
int balance = 10;
while(true){
 if(balance < 9)
   break;
 balance = balance - 9;
}
System.out.println(“Balance is “ + balance);
```

JAVA OVERVIEW

- What is the following Java code?

```
int balance = 10;
while(true){
 if(balance < 9)
   continue;
 balance = balance - 9;
}
System.out.println("Balance is " + balance);
```

Input and Output

- **`Scanner`** object to read from the keyboard (`System.in`)
- **`PrintWriter`** object to write to the screen (`System.out`)

Methods

- Block of java code with inputs and one output (or none)
- Inputs: List of parameters (arguments)
- Output: return value (or void)
- Can be called several times
- Arguments are passed by value

Arrays

- Collection of variables of the same type
- 1D array (one index)
- 2D array (two indices)
- Multi-dimensional array (n indices)
- Array arguments are passed by reference

JAVA OVERVIEW

- What is the output of the following Java code?

```
public class Test {
  public static void main(String[] args){
      int[] list = {1, 2, 3, 4, 5};
      doSomething(list);
      for(int i = 0; i < list.length; i++)
        System.out.print(list[i] + " “);
  }
  public static void doSomething(int[] in){
      for(int i = 0; i < in.length/2; i++) {
        in[i] = in[in.length - i - 1];
  }
}
```

JAVA OVERVIEW

- What is the output of the following Java code?

```
public class Test {
  public static void main(String[] args){
      int[] list = {1,2,3, 4, 5};
      doSomething(list);
      for(int i = 0; i < list.length; i++)
        System.out.print(list[i] + " “);
  }
  public static void doSomething(int[] in){
      int[] out = new int[in.length];
      for(int i = 0; i < in.length; i++) {
        out[i] = in[in.length - i - 1];
      }
      in = out;
  }
}
```

Command-Line Arguments

- Arguments passed to the main function
- List of arguments are stored in the array `String[] args` (parameter of the main method)

JAVA OVERVIEW

- What is the output of the following Java code if it is run using the following command: "***java Test 2 3***"?

```
public class Test {
   public static void main(String[] args){
     if(args.length !=3){
       System.out.println("Three arguments must be provided.");
       System.exit(1);
     }
     int op1, op2, result = 0;
     op1 = Integer.parseInt(args[0]);
     op2 = Integer.parseInt(args[2]);
     switch(args[1].charAt(0)){
       case '+': result = op1 + op2; break;
       case '-': result = op1 - op2; break;
       case '*': result = op1 * op2; break;
       case '/': result = op1 / op2; break;
      default: System.out.println("Invalid operator.");
     }
     System.out.println(args[0]+' '+args[1]+' '+args[2]+ " = "+result);
   }
}
```

JAVA OVERVIEW

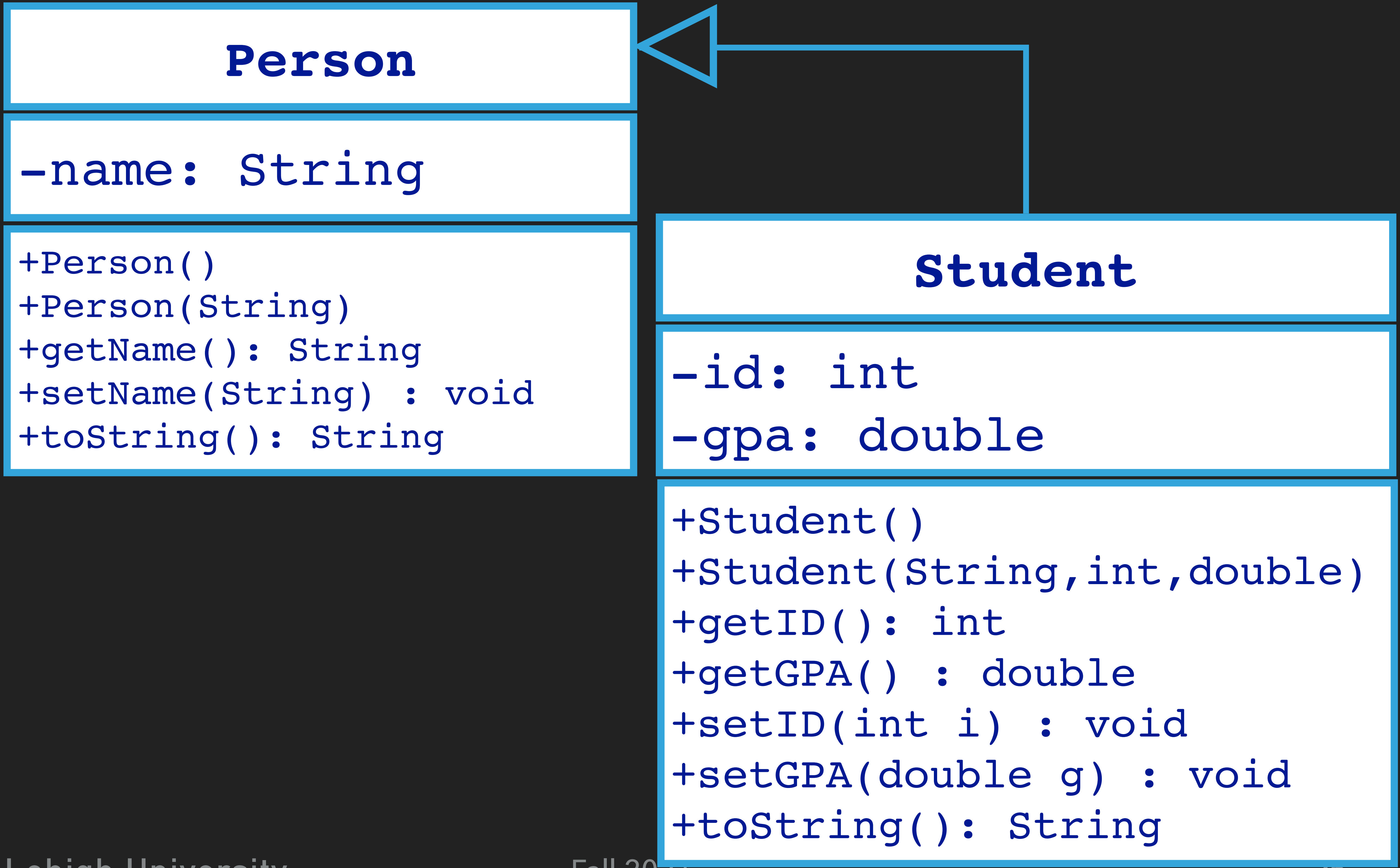
- What is the output of the following Java code if it is run using the following command: "**java Test 2 + 3**"?

```
public class Test {
   public static void main(String[] args){
     if(args.length !=3){
       System.out.println("Three arguments must be provided.");
       System.exit(1);
     }
     int op1, op2, result = 0;
     op1 = Integer.parseInt(args[0]);op1=2;
     op2 = Integer.parseInt(args[2]);op2=3;
     switch(args[1].charAt(0)){'+'
       case '+': result = op1 + op2; break; result=5
       case '-': result = op1 - op2; break;
       case '*': result = op1 * op2; break;
       case '/': result = op1 / op2; break;
      default: System.out.println("Invalid operator.");
     }
     System.out.println(args[0]+' '+args[1]+' '+args[2]+ " = "+result);
   }
}
```

Object Oriented Programming

- Create classes - programmer created types
- Create objects - instantiate the classes
- Create new classes by extending existing classes - inheritance
- Use the super class type to hold instances of the sub classes - polymorphism

Practice


Person
-name: String
+Person()
+Person(String)
+getName(): String
+setName(String) : void
+toString(): String
Student
-id: int
-gpa: double
+Student()
+Student(String,int,double)
+getID(): int
+getGPA() : double
+setID(int i) : void
+setGPA(double g) : void
+toString(): String

Object Oriented Programming

```java
// Class Person
public class Person {
   private String name;
   // default constructor
   public Person() {
      name="";
   }
   // Constructor with one parameter
   public Person(String name) {
      this.name = name;
   }
   // Accessor (getter)
   public String toString() {
      return name;
   }
   // Mutator (setter)
   public void setName(String name) {
      this.name = name;
   }
}
```

Object Oriented Programming

```
// Class Student inherits class Person
public class Student extends Person{
   private int id;
   private double gpa;
   // default constructor
   public Student() {
      super(); id=0; gpa=0.0;
   }
   // Constructor with three parameters
   public Student(String name, int id, double gpa) {
      super(name); this.id = id; this.gpa = gpa;
   }
   // Accessors (getters)
   public int getID() { return id;}
   public double getGPA() { return gpa;}
   public String toString() {
      return super.toString() + "\t" + id + "\t" + gpa;
   }
   // Mutators (setters)
   public void setID(int id) { this.id = id;}
   public void setGPA(double gpa) { this.gpa = gpa;}

}
```

Object Oriented Programming

```
// Class TestStudent
import java.util.Scanner;
public class TestStudent {
  public static void main(String[] args) {
   Scanner input = new Scanner(System.in);
   System.out.println("Enter the number of students: ");
   int studentCount = input.nextInt();
  // Creating an array studentList (type Person)
   Person[] studentList = new Person[studentCount];
   for(int i=0; i<studentCount; i++) {
     String name; int id; double gpa;
     System.out.println("Enter student information" +
                                "(name id gpa): ");
     name = input.next() + input.next();
     id = input.nextInt();
     gpa = input.nextDouble();
      // Creating instances of class Student
     studentList[i] = new Student(name, id, gpa);//polymorphism
   }
   printArray(studentList);
 }}
```

Object Oriented Programming

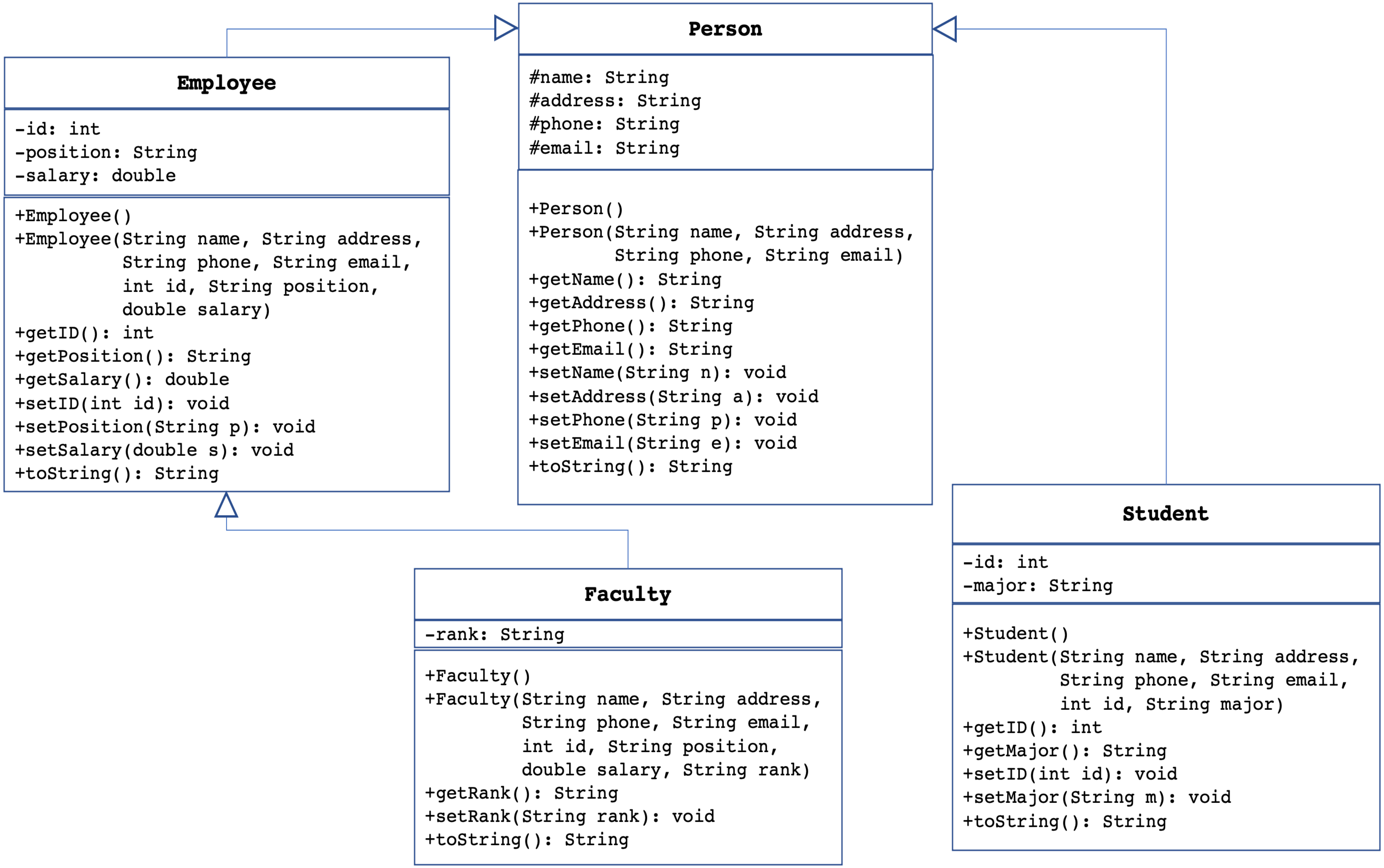
```
// Definition of the method printArray()
 public static void printArray(Person[] list) {
  for (int i=0; i<list.length; i++)
    System.out.println(list[i].toString());
 }
```

Practice

Analyze the given UML diagram

- ✦ Describe the relationships between the classes `Person`/`Student`/`Employee`/`Faculty`
- ✦ How many data/method members are in the classes `Person`/`Student`/`Employee`/`Faculty`? What is the access modifier of each member of these classes? Which methods are accessors/mutators?

Practice



```
Person
#name: String
#address: String
#phone: String
#email: String

+Person()
+Person(String name, String address,
        String phone, String email)
+getName(): String
+getAddress(): String
+getPhone(): String
+getEmail(): String
+setName(String n): void
+setAddress(String a): void
+setPhone(String p): void
+setEmail(String e): void
+toString(): String
```

```
Employee
-id: int
-position: String
-salary: double

+Employee()
+Employee(String name, String address,
          String phone, String email,
          int id, String position,
          double salary)
+getID(): int
+getPosition(): String
+getSalary(): double
+setID(int id): void
+setPosition(String p): void
+setSalary(double s): void
+toString(): String
```

```
Faculty
-rank: String

+Faculty()
+Faculty(String name, String address,
         String phone, String email,
         int id, String position,
         double salary, String rank)
+getRank(): String
+setRank(String rank): void
+toString(): String
```

```
Student
-id: int
-major: String

+Student()
+Student(String name, String address,
         String phone, String email,
         int id, String major)
+getID(): int
+getMajor(): String
+setID(int id): void
+setMajor(String m): void
+toString(): String
```

Instance/Static Members

Identify the statements that are correct (right code)

```
public class Item {
 // instance variable
 private int i;
 //static variable
 public static String s;
 //instance method
 public void iMethod()
 {}
 //static method
 public static void sMethod()
 {}
}
```

```
Item i1 = new Item();
System.out.println(i1.i);
System.out.println(i1.s);
i1.iMethod();
i1.sMethod();
System.out.println(Item.i);
System.out.println(Item.s);
Item.iMethod();
Item.sMethod();
```

Passing Objects to Methods

Show the output of the following program

```
public class Counter {

 private int count;
 public Counter()
 { count = 1;}

 public Counter(int c)
 { count = c; }

 public int getCount()
 { return count;}

 public void increment()
 { count++; }
}
```

```
public static void main…
{
  int times = 0;
  Counter myCounter = new Counter();
  for(int i=0;i<100;i++)
   update(myCounter, times);
  System.out.println("Count is " +
                     myCounter.getCount());
  System.out.println("times is " +
                                 times);
}
public static void update(Counter c,
                                int t){
  c.increment();
  t++;
}
```

IDEs

- Integrated Development Environment
  - Write, Compile, Execute Java code
- Visual Studio Code, Eclipse, NetBeans
- Available free for download at eclipse.org , netbeans.org, code.visualstudio.com

NEXT SESSION

- Active Learning Activity
  - Using an IDE
  - Using a version control system (have git installed on your computer and register in github.com)
  - Implement the class hierarchy shown in the UML diagram (slide 41)