PROGRAMMING AND DATA STRUCTURES

INTRODUCTION

HOURIA OUDGHIRI

FALL 2023

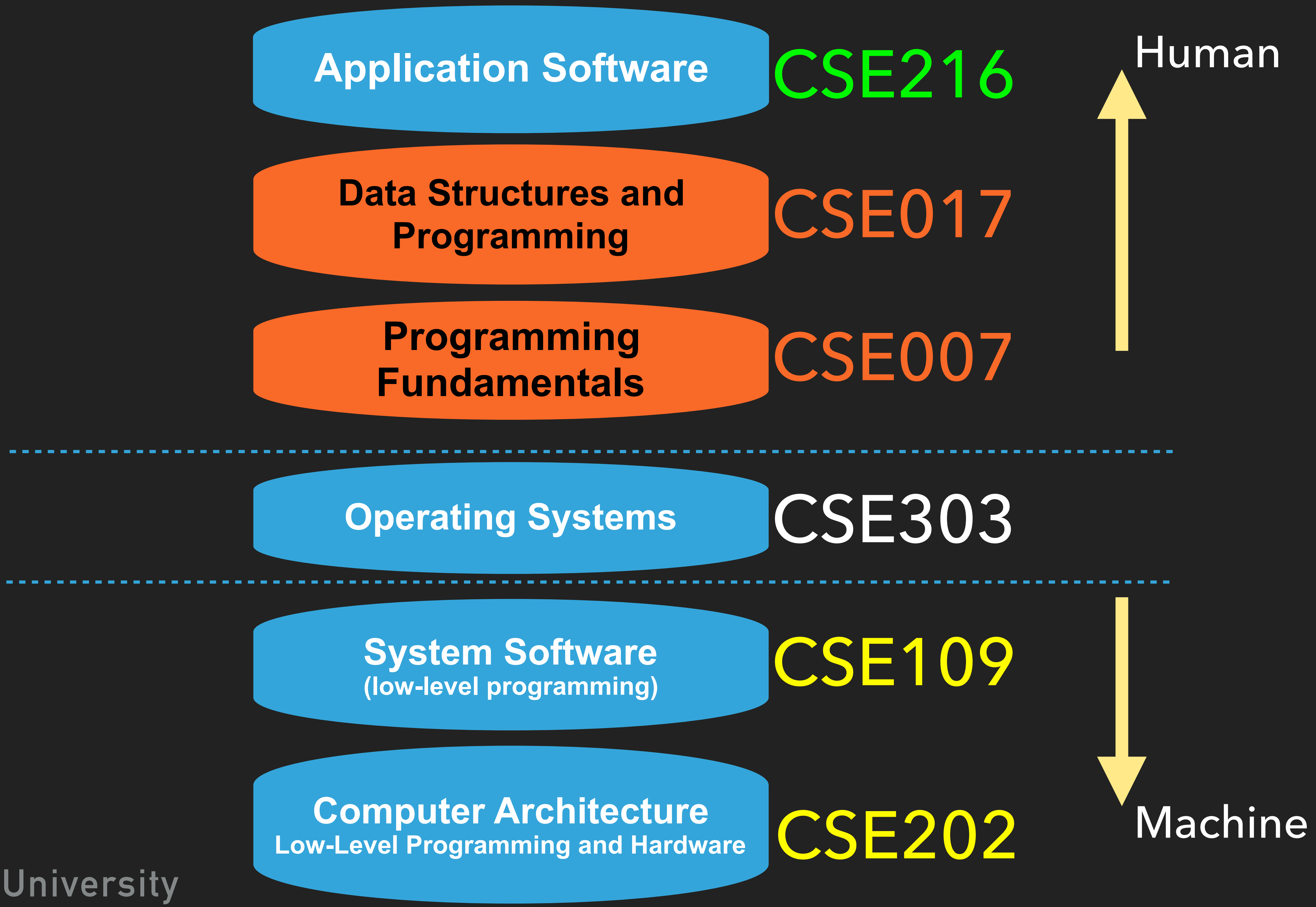
OUTLINE

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

WHAT IS CSE017?

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

WHERE IS CSE017?



WHAT IS IN CSE017?

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

STUDENT LEARNING OUTCOMES

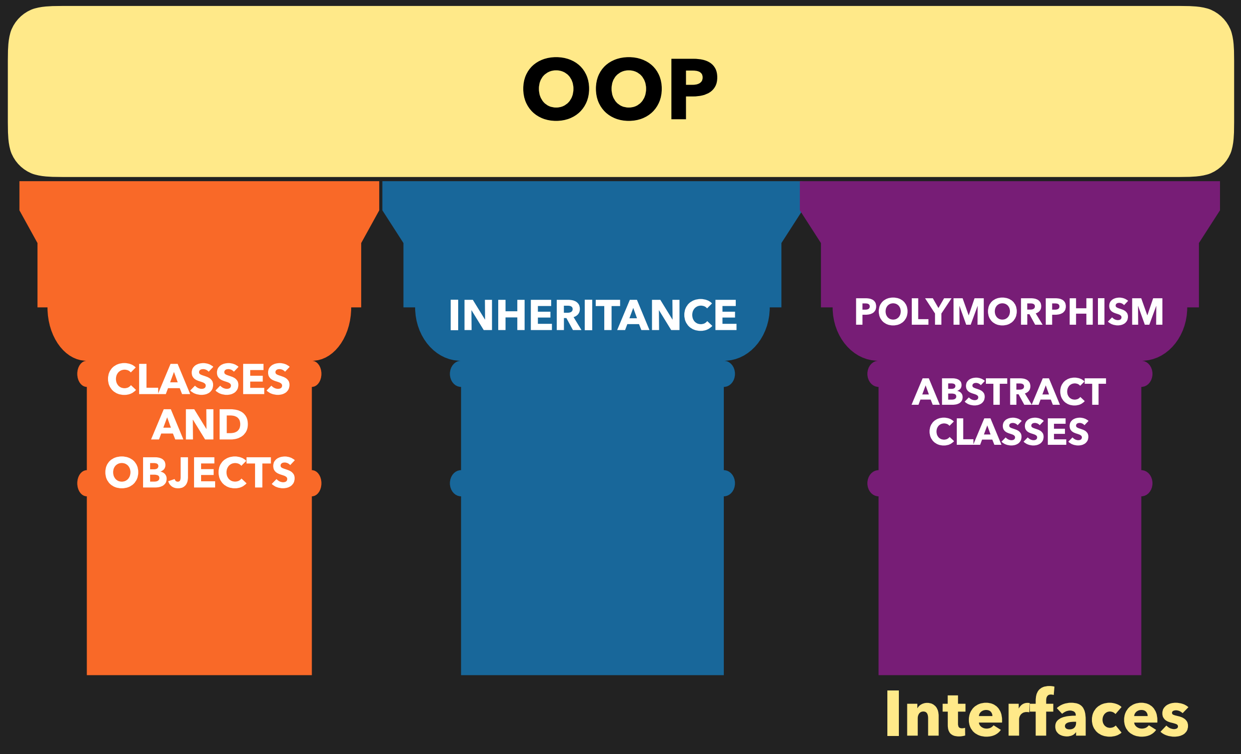
- What knowledge and skills would you acquire by the end of the course?

STUDENT LEARNING OUTCOMES

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

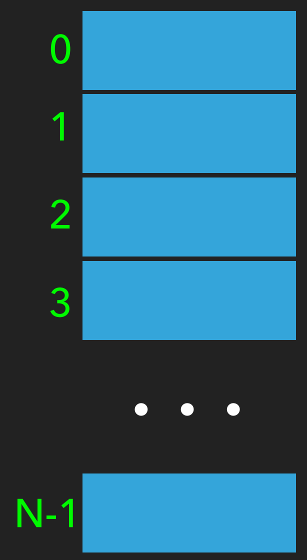
STUDENT LEARNING OUTCOMES

1. Apply Object Oriented Concepts to write Java programs

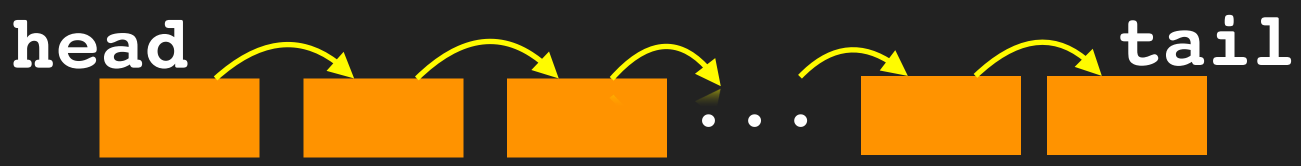


STUDENT LEARNING OUTCOMES

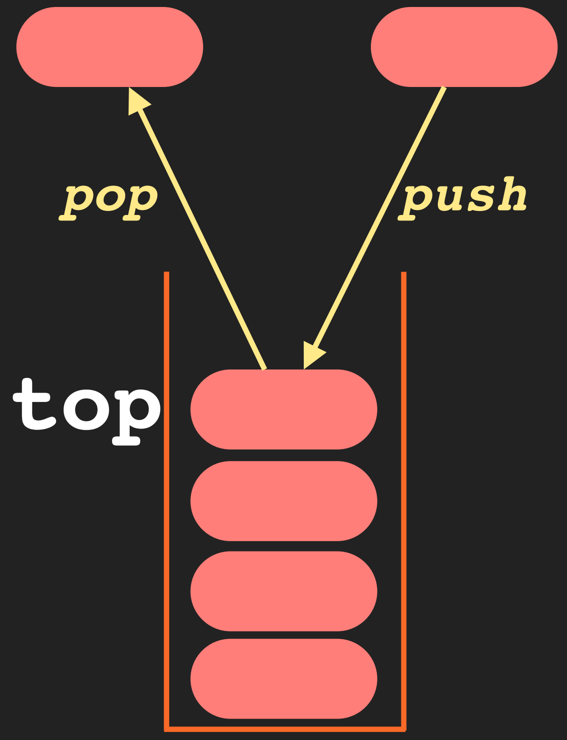
2. Implement common data structures to store and manipulate data



Array List



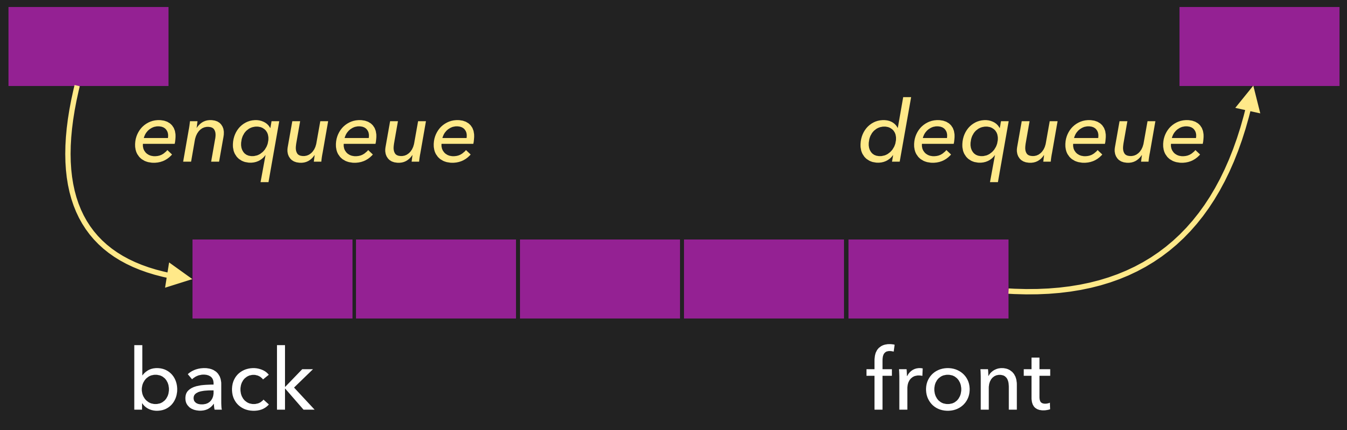
Linked List



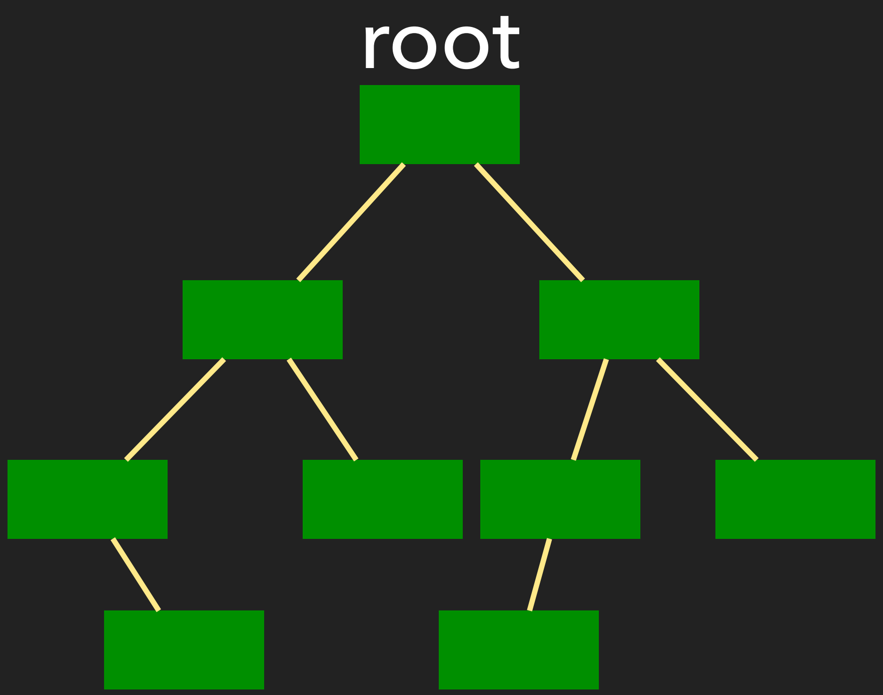
Stack

STUDENT LEARNING OUTCOMES

2. Implement common data structures to store and manipulate data



Queue

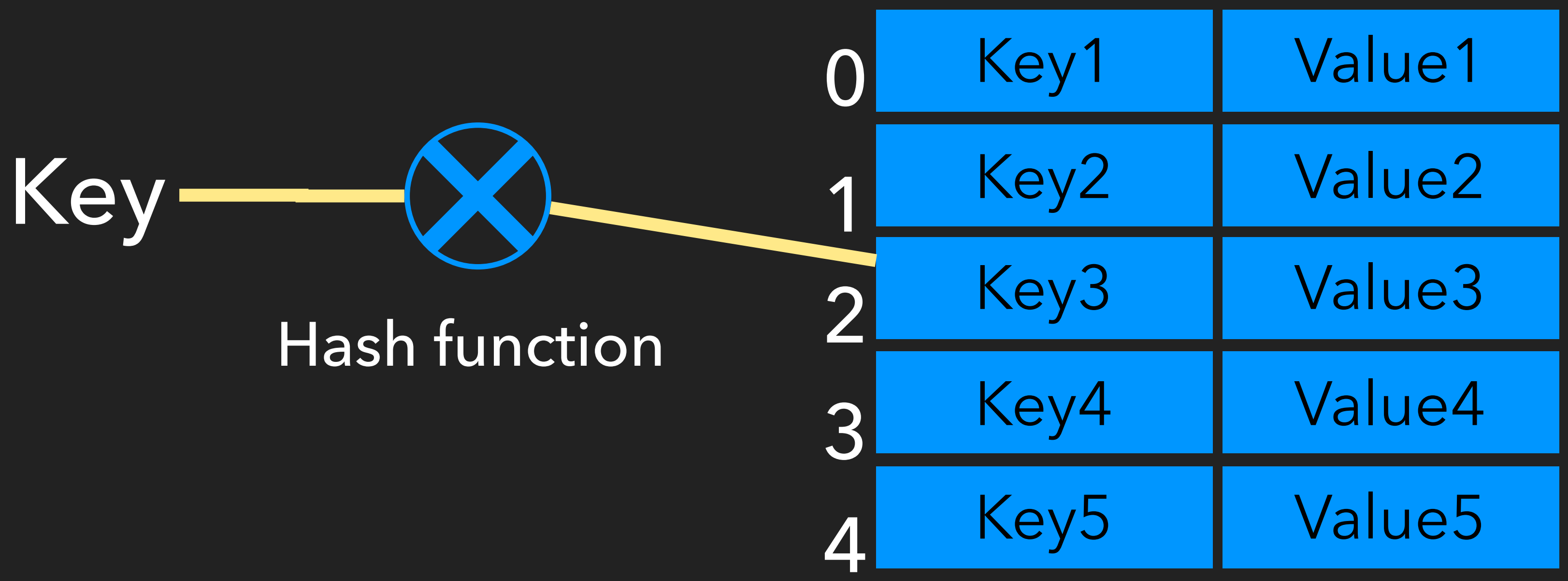


Tree

Tree

STUDENT LEARNING OUTCOMES

2. Implement common data structures to store and manipulate data



HashTable (HashMap)

Tree

STUDENT LEARNING OUTCOMES

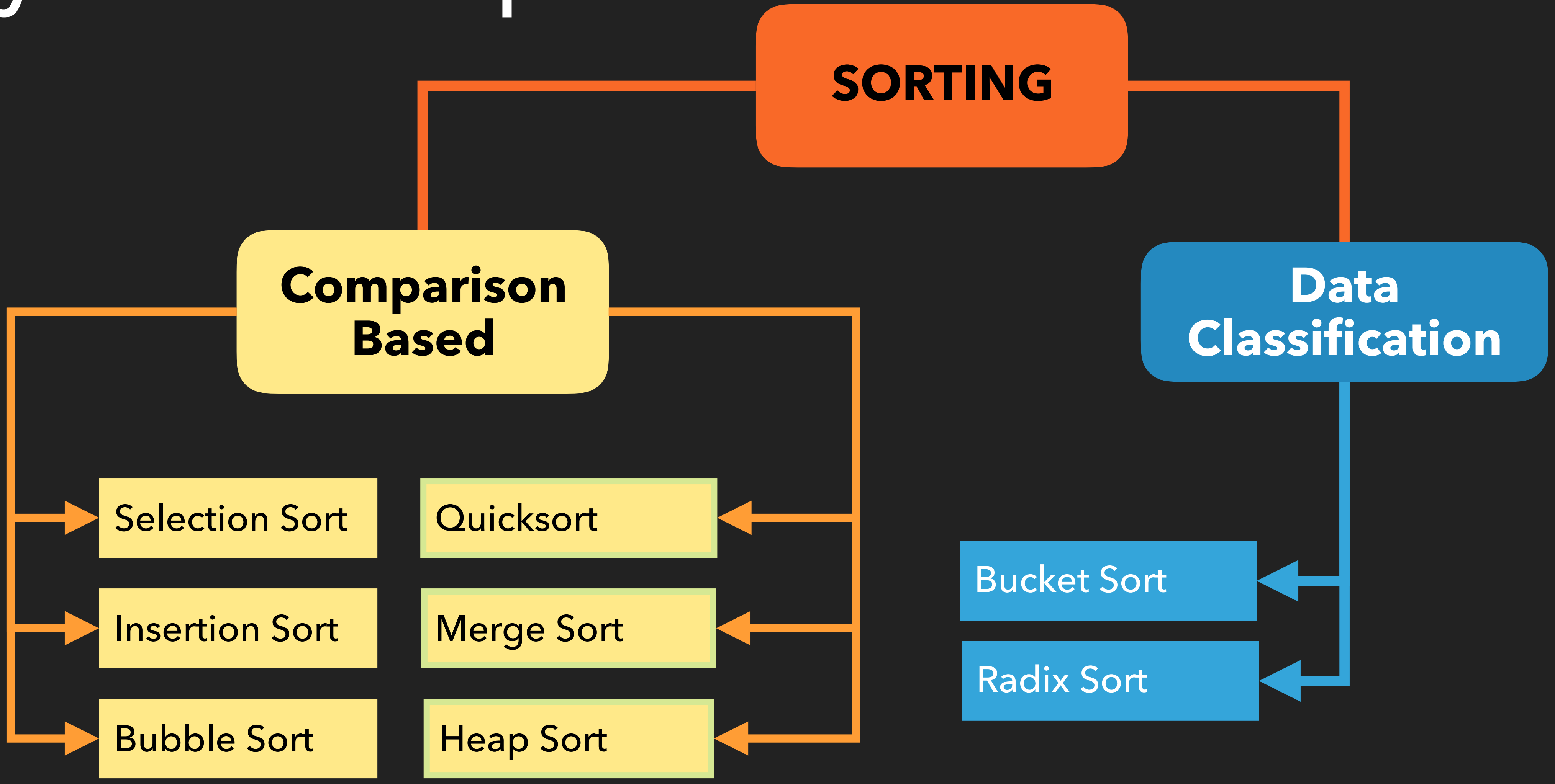
3. Use recursion to implement algorithms

```java
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;
 }
}
```

```java
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 (keyboard, command-line) and Output (console)
- 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 - `?:`

JAVA OVERVIEW

What is the output of the following Java code for `x1 = 8` and `x2 = 12`?

```
public class Test {
  public static void main(String[] args) {
    int x1, x2, score, scale = 10;
    java.util.Scanner input = new java.util.Scanner(System.in);

    x1 = input.nextInt();
    score = (x1 > 10) ? 3 * scale : 4 * scale;
    System.out.println("score = " + score);

    x2 = input.nextInt();
    System.out.println((x2 % 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 = 85.5`?

```java
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 codes?

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

```java
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`)
- Command-line arguments to the main function (the array **`args`**)
- **`PrintWriter`** object to write to the console (`System.out`)

Input and Output

What is the output of the following Java code?

```
public class InputOutput{
    public static void main(String[] args){
        int number1 = Integer.parseInt(args[0]);
        int number2 = Integer.parseInt(args[1]);
        System.out.println(number1 + " * " + number2 +
                             " = " + (number1 * number2));
    }
}
```

```
> javac InputOutput.java
> java InputOutput 12 5
```

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 of primitive type 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)
- Arrays are passed by reference

What is the output of the following Java code?

```
public class class_code {

    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++) {
            int temp = in[i];
            in[i] = in[in.length - i - 1];
            in[in.length-i-1] = temp;
        }
    }
}
```

JAVA OVERVIEW

What is the output of the following Java code?

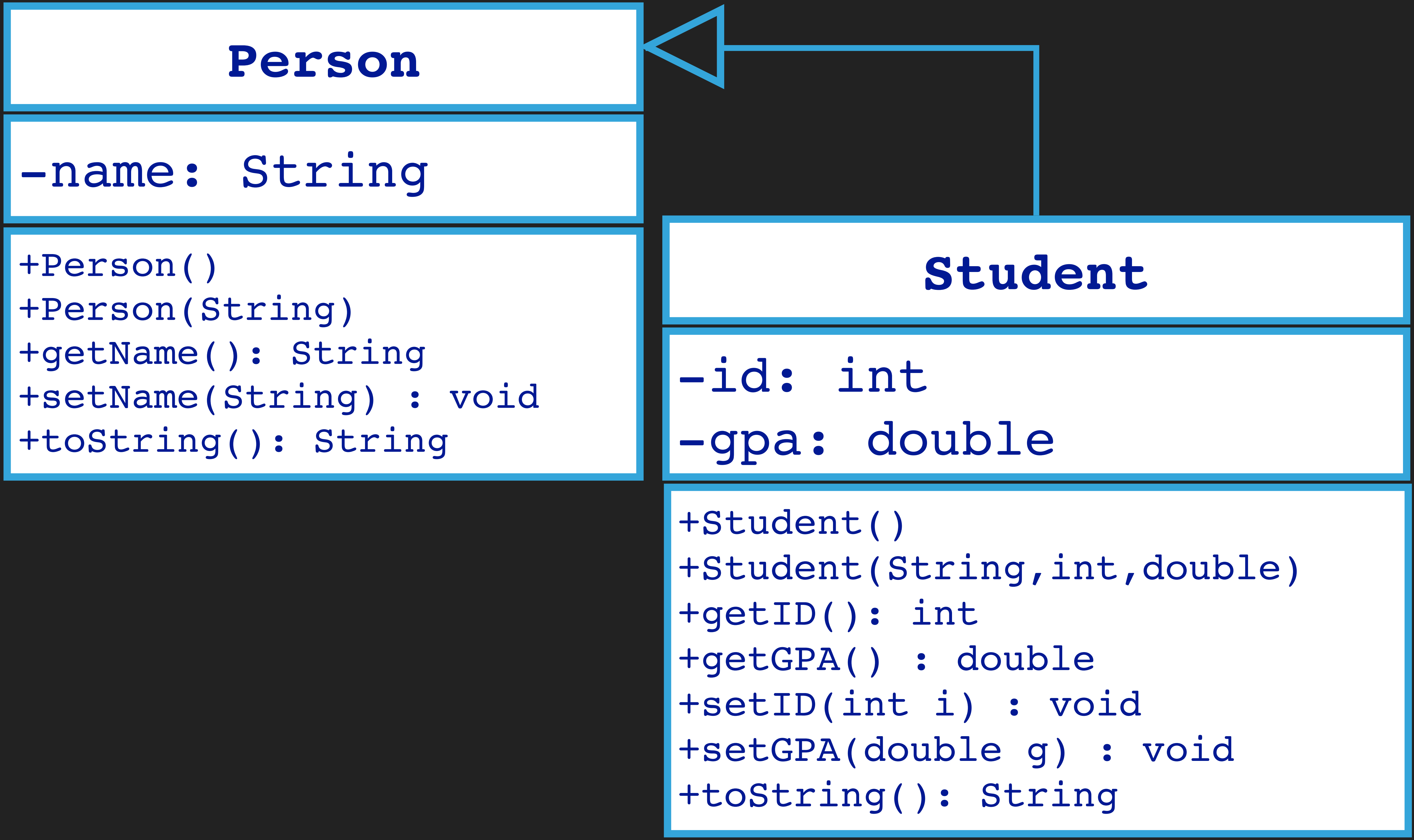
```
public class Test {

    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 int[] 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;
    }
}
```

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

Object Oriented Programming



Object Oriented Programming

```
// Class Person
public class Person {
  private String name;
  // default constructor
  public Person() {
    this("none");
  }
  // 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

```
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 the class Student
    studentList[i] = new Student(name, id, gpa);//polymorphism
   }
   printArray(studentList);
 }
}
```

Object Oriented Programming

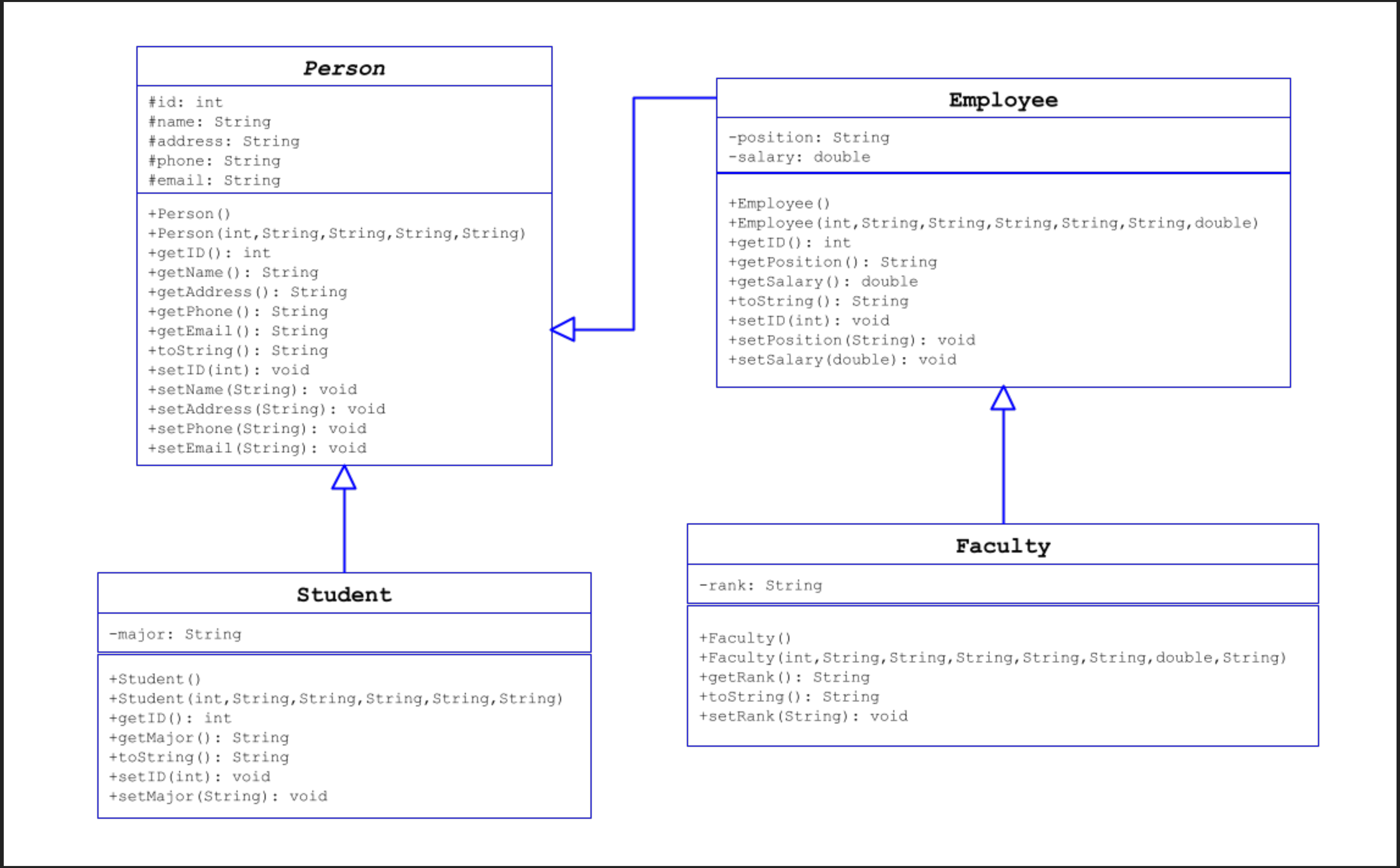
```
// Definition of the method printArray()
 public static void printArray(Person[] list) {
  for (int i=0; i<list.length; i++){
     System.out.println(list[i].toString());
  }
 }
```

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 in these classes?
- ✦ Which methods are accessors/mutators?

Practice


Person
#id: int
#name: String
#address: String
#phone: String
#email: String
+Person()
+Person(int,String,String,String,String)
+getID(): int
+getName(): String
+getAddress(): String
+getPhone(): String
+getEmail(): String
+toString(): String
+setID(int): void
+setName(String): void
+setAddress(String): void
+setPhone(String): void
+setEmail(String): void
Employee
-position: String
-salary: double
+Employee()
+Employee(int,String,String,String,String,String,double)
+getID(): int
+getPosition(): String
+getSalary(): double
+toString(): String
+setID(int): void
+setPosition(String): void
+setSalary(double): void
Student
-major: String
+Student()
+Student(int,String,String,String,String,String)
+getID(): int
+getMajor(): String
+toString(): String
+setID(int): void
+setMajor(String): void
Faculty
-rank: String
+Faculty()
+Faculty(int,String,String,String,String,String,double,String)
+getRank(): String
+toString(): String
+setRank(String): void

Instance/Static Members

Identify the statements that are correct (right code)

```
public class A {
 // instance variable
 private int iv;

 // static variable
 public static String sv;

 // instance method
 public void iMethod()
 {}

 // static method
 public static void sMethod()
 {}
}
```

```
public class Test{
 public static void main(String[] args){
    A a1 = new A();

    // Using instance a1 of class A
    System.out.println(a1.iv);
    System.out.println(a1.sv);
    a1.iMethod();
    a1.sMethod();

    // Using the class name A
    System.out.println(A.iv);
    System.out.println(A.sv);
    A.iMethod();
    A.sMethod();
 }
}
```

Passing Objects to Methods

Show the output of the following program

```java
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++; }
}
```

```java
public class Test{
 public static void main(String[] args){
    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++;
  }
}
```

IDE

- Integrated Development Environment
  - Write, Compile, Execute Java code
- Visual Studio Code with remote SSH extension (work remotely on Sunlab machines)
- Available free for download code.visualstudio.com

NEXT CLASS

- Active Learning Activity #1
  - Use an IDE
  - Use a version control system (register on github.com, use your Lehigh email)
  - Implement the class hierarchy shown in the UML diagram (slide 35)