

CS 1027

Fundamentals of Computer
Science II

Course Introduction

Lecturers

Ahmed Ibrahim(Section 001)

Roberto Solis-Oba (Section 002)



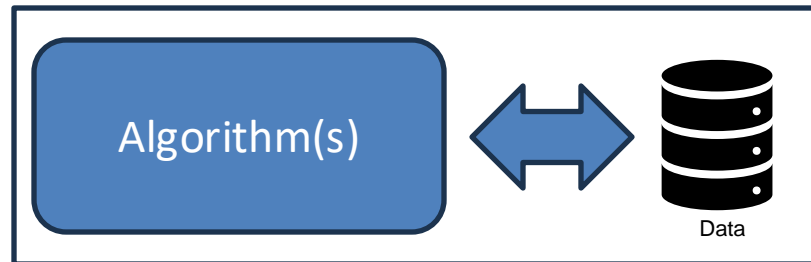
What is CS1027 about?

- The design of algorithms and their implementation in an Object-Oriented Programming environment
- The organization and manipulation of data

Computer Programs

- A computer program consists of 2 parts:
 - Algorithms – Set of instructions that complete a task in a finite amount of time
 - Data Structures – Ways to organize data in memory

A program:



Example

```
1  public class SumArray {  
2  |  
3  public static void main(String[] args) {  
4      int[] numbers = {5, 10, 15, 20, 25};  
5      int sum = 0;  
6  
7      for (int number : numbers) {  
8          sum += number;  
9      }  
10  
11     System.out.println("The sum of the array is: " + sum);  
12 }  
13 }
```

Run | Debug

Starting point

Data

Algorithm

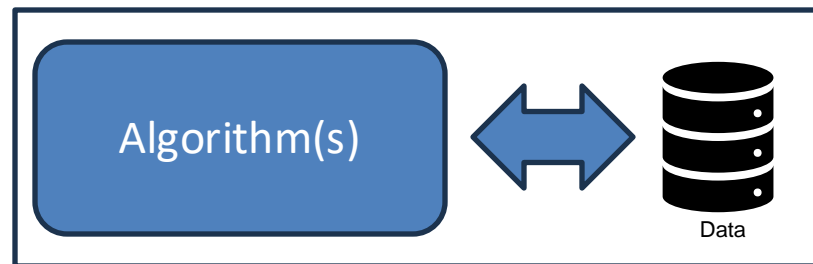
Output

Computer Programs (cont.)

Computer Programs must be:

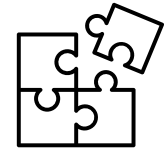
- **Correct**– A program must always terminate and produce the correct output
- **Efficient**– A program must require a small amount of time and memory

A program:

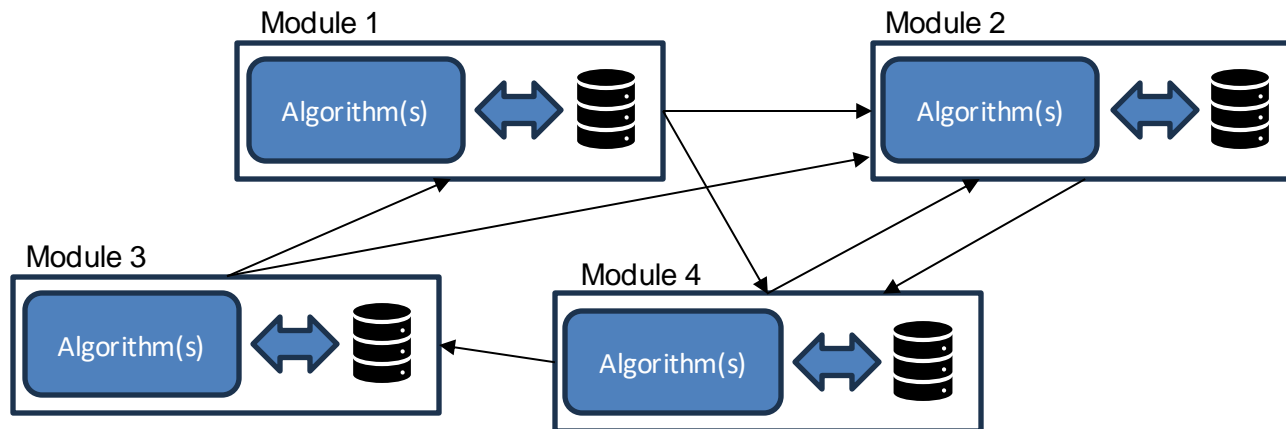


Algorithm Flow

Computer Programs



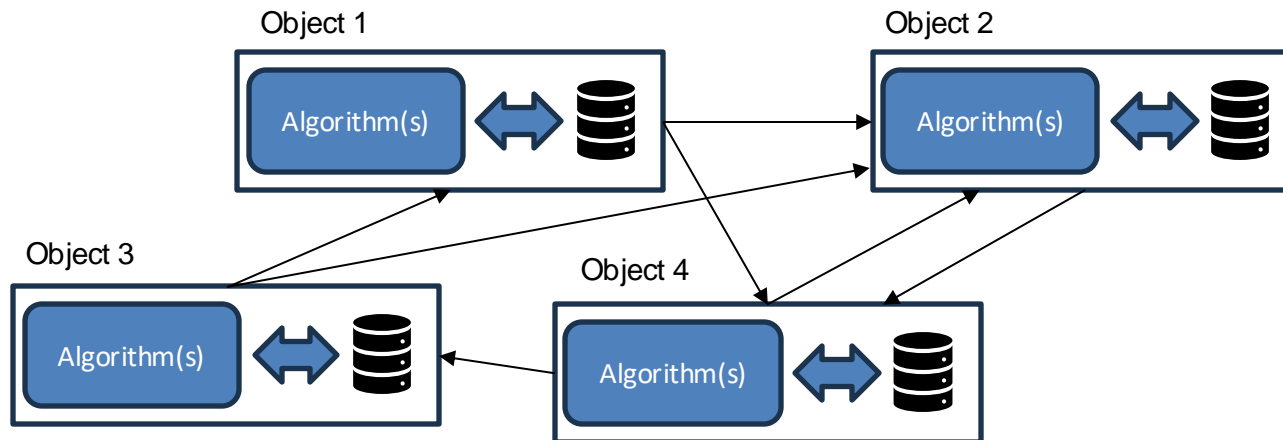
- A computer program is divided into pieces or modules, each with its own algorithms and data, to simplify its design.
- Each module implements a **specific functionality**, and modules interact with each other.



Hypothetical Modular System

Object Oriented Methodology

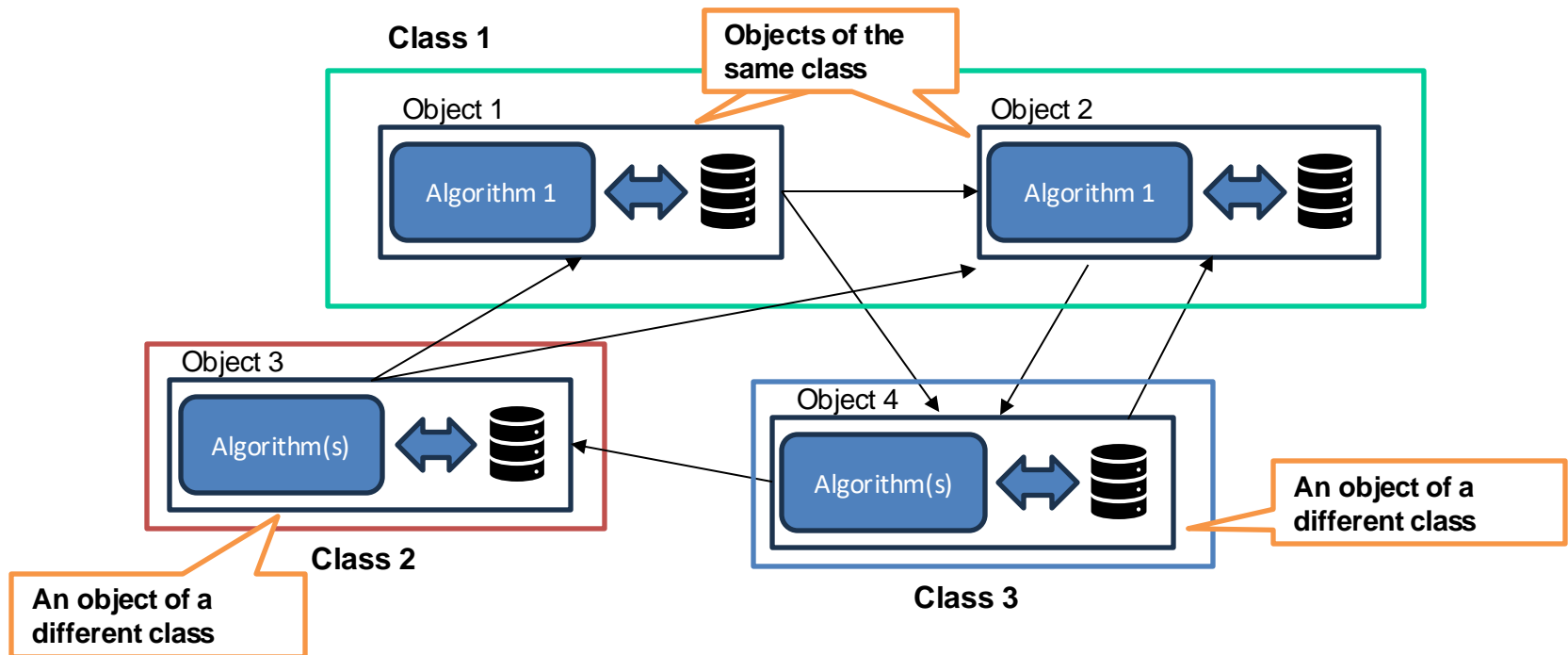
- In Object-Oriented, these modules are called objects.



Object-Oriented Design

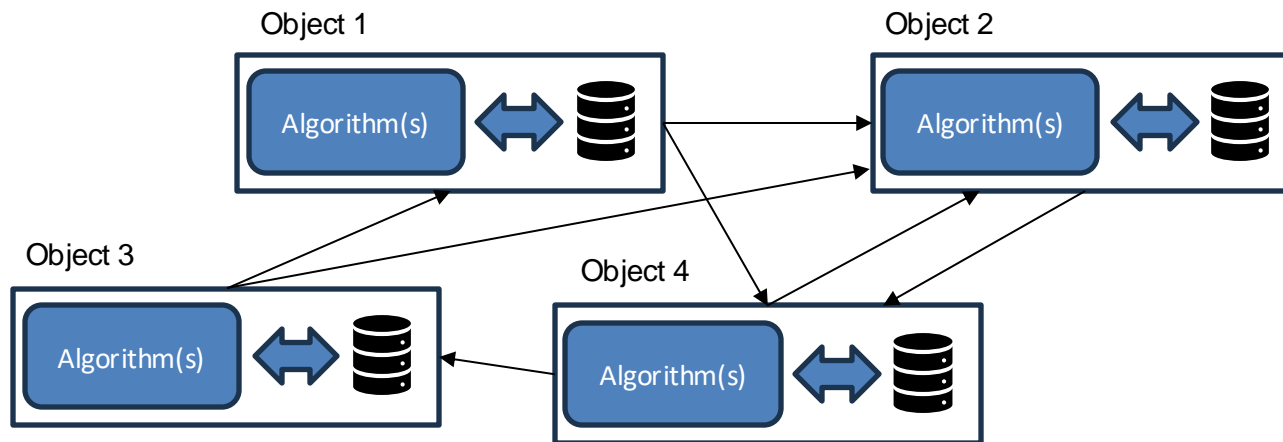
Object Oriented Methodology

- Objects belong to classes. Two objects of the **same class** have the same algorithms and the same type of data.



Object Oriented Design (OOD) Principles

- OO design principles, such as **Encapsulation**, **Abstraction**, **Inheritance**, and **Polymorphism**, aim to simplify the task of designing correct programs.



Object-Oriented Design

Course Topics

- Algorithms and data structures
- Object-oriented design concepts: encapsulation, abstraction, inheritance, polymorphism
- Exceptions and debugging
- Arrays and linked structures
- Abstract data types and their implementations: stacks, queues, lists, trees
- Recursion and memory management
- Sorting algorithms

Learning Outcomes

Upon completion of this course, you should be able to:

1. Program in an object-oriented language (*Java*)
2. Create classes and use inheritance for sub-classes
3. Identify arrays and linked data structures
4. Implement and use abstract data types: stacks, queues, lists, and trees
5. Debug code and use exceptions
6. Identify the different parts of computer memory during the execution of a Java program.
7. Design algorithms and pseudo-code and implement them

Computing Environment

- Programming will be done in *Java*
- The recommended IDE for CS1027 is *Eclipse*.
 - You may use a different IDE if you prefer, but keep in mind that the instructors and TAs might not be familiar with the IDE you choose.
 - Ensure you have JDK (Java Development Kit) and JVM (Java Virtual Machine) installed on your machine.
 - Check www.oracle.com/java.

Computing Environment

- Prerequisites: **1025A/B or CS 1026A/B** (ES 1036A/B engineering students) or DS 1200A/B, with a mark of at least 65% in either
- "Unless you have either the prerequisite for this course or written special permission from your Dean to enroll in it, you will be removed from this course, and it will be deleted from your record. This decision may not be appealed. You will receive no adjustment to your fees." Instructor permission is also acceptable.
- You must be comfortable with programming syntax in Python or Java.
Note: We will not teach Java syntax. We will cover some of the language's most complex features, but you are expected to learn Java on your own.

ZyBooks

(Recommended)

1. Sign in or create an account at
<http://learn.zybooks.com>

2. Enter zyBook code:
UWOCOMPSCI1027A-BSolis-ObaFall2024
3. Subscribe

OR pay through the Bookstore for your ZyBooks code.

CS 1027 Website

- OWL Brightspace site: "[COMPSCI 1027A](#)
[COMPUTER SCIENCE FUNDAMENTALS II](#)"
- Course-related information:
 - Lecture notes
 - Lab instructions
 - Assignments
 - Sample code
- Check it frequently for announcements and updated course material

Lecture Notes

- Available on the course OWL Brightspace site
- They are intended to help in note-taking during lectures
- They are **NOT** a substitute for attending lectures
- There may be other material presented in lectures also.

Labs

- 1 lab hour per week
- Labs start the week of **September 16-20**
- Purpose of labs: to introduce or expand on practical material + programming exercises
- Lab instructions will be posted on the course site
 - Read through the lab instructions before going to the lab.
 - Do the pre-lab preparation.
 - **No makeup Labs**
 - You **must** attend your lab in-person and show your work to the TA in charge of your section. You cannot do the lab from home – you will not get the mark for it.
 - You must also submit your lab code through OWL Brightspace to get credit for a lab.

Labs (cont.)

- The lowest two labs will be dropped, so you can miss up to two labs without penalty. However, you are strongly encouraged to complete all the labs.
- If you have to miss a lab or two, **do not attend a different lab section. That is not allowed.** You must attend the section in which you are registered. We drop the lowest 2, so it's ok to miss one or two labs.
- Remember that **attendance is mandatory** to get the mark for the lab, so **do not do the lab work at home.** You are not allowed to show up with the work already done and expect to show the TA and leave. The work must be done during the lab session in which you are registered.

Email Contact

- *Email from instructors to you:*
 - Course email will be sent to your *uwo* email accounts
 - You are responsible for information sent via email to your account
 - OWL announcements are usually accompanied with an email to your uwo account

Email Contact

- *Email from you to TAs/instructors:*
 - Feel free to email with *brief* questions related to lecture material or clarification of assignments
 - You must send an email from your *uwo* account
 - **You must** include “CS1027” in the Subject line
 - Instructors usually do not read emails at night or during weekends!

Student Evaluation

	Weight	Due Date (tentative)
Assignment 1	8%	October 8
Assignment 2	8%	October 31
Assignment 3	8%	November 18
Assignment 4	8%	December 5
Labs	8%	Weekly
Midterm Exam	20%	October 26
Final Exam	40%	TBA

- All assignments are due at 11:55 pm.

Important Conditions

To pass the course:

- **Weighted average of exams must be at least 45%**
- Weighted average of assignments must be at least 45%
- Otherwise, your maximum course grade is 45%

To achieve a final grade of 65% or higher:

- **Weighted average of exams must be at least 50%**
- Weighted average of assignments must be at least 50%
- Otherwise, your maximum course grade is 60%

Student Evaluation

- If an assignment is canceled for any reason, the other assignments will be adjusted so that they still add up to 32%.
- If the midterm exam is canceled for any reason, its weight will be shifted to the final exam.

Midterm Exam

- This is a 2-hour midterm exam tentatively scheduled for **Saturday, October 26**, but the university still needs to confirm it.
- **There is NO MAKEUP exam for this course.**
- If you cannot write the midterm exam for a valid reason, you must contact *Academic Counselling* and request an **academic consideration**. If granted, the weight of the midterm exam will be shifted to the final exam.

Programming Assignments

- Assignment Submission:
 - Details will be posted on OWL Brightspace
 - Submit assignments on ***Gradescope***
- Late Assignments:
 - The late penalty is 10% of the max. assignment mark per day late
 - Maximum two days late
 - No extensions are given unless you have accommodations

Assignment Marking

- A large portion of your assignment grade will come from auto-graded tests
 - You must pass the tests on *Gradescope* to earn those marks.
- The rest comes from additional tests, code logic, formatting, comments, and following good programming practices.
- **One week** time limit on requesting adjustment in an assignment mark

Assignment Marking

- For questions regarding an assignment mark, you must first contact and discuss your concerns with your Teaching Assistant.
- If the matter remains unresolved, you may then take your concerns to your course instructor.
- Keep a duplicate copy of all your assignments, just in case...

Assignment Marking

- Assignments are to be completed by **individuals**, not pairs or groups.
- Collaboration that results in assignments that are more than coincidentally alike is unacceptable and will be regarded as an occurrence of academic dishonesty.
- We have software in place that will examine your code against everyone else in both sections and report any incidents of copying or sharing.

What is academic dishonesty?

- Collaboration
- Copying another student's assignment
- Allowing another student to copy
- Using code from books or the Internet
- Paying someone to write your code
- Using ChatGPT or another AI system for all or even a portion of your work
- **Penalty for academic dishonesty:** reported to the Dean, and you will receive a mark of 0%.

CS 1026 uses **Python**, so why are we switching to **Java**?

- Many different programming languages (C, C++, C#, Pascal, Modula 2, Turing, Logo, Lisp, Prolog, Oberon, Ruby, ...) are designed for a particular purpose.
- Programmers should know several programming languages to select the most appropriate one for a particular task.



CS 1026 uses **Python**, so why are we switching to **Java**?

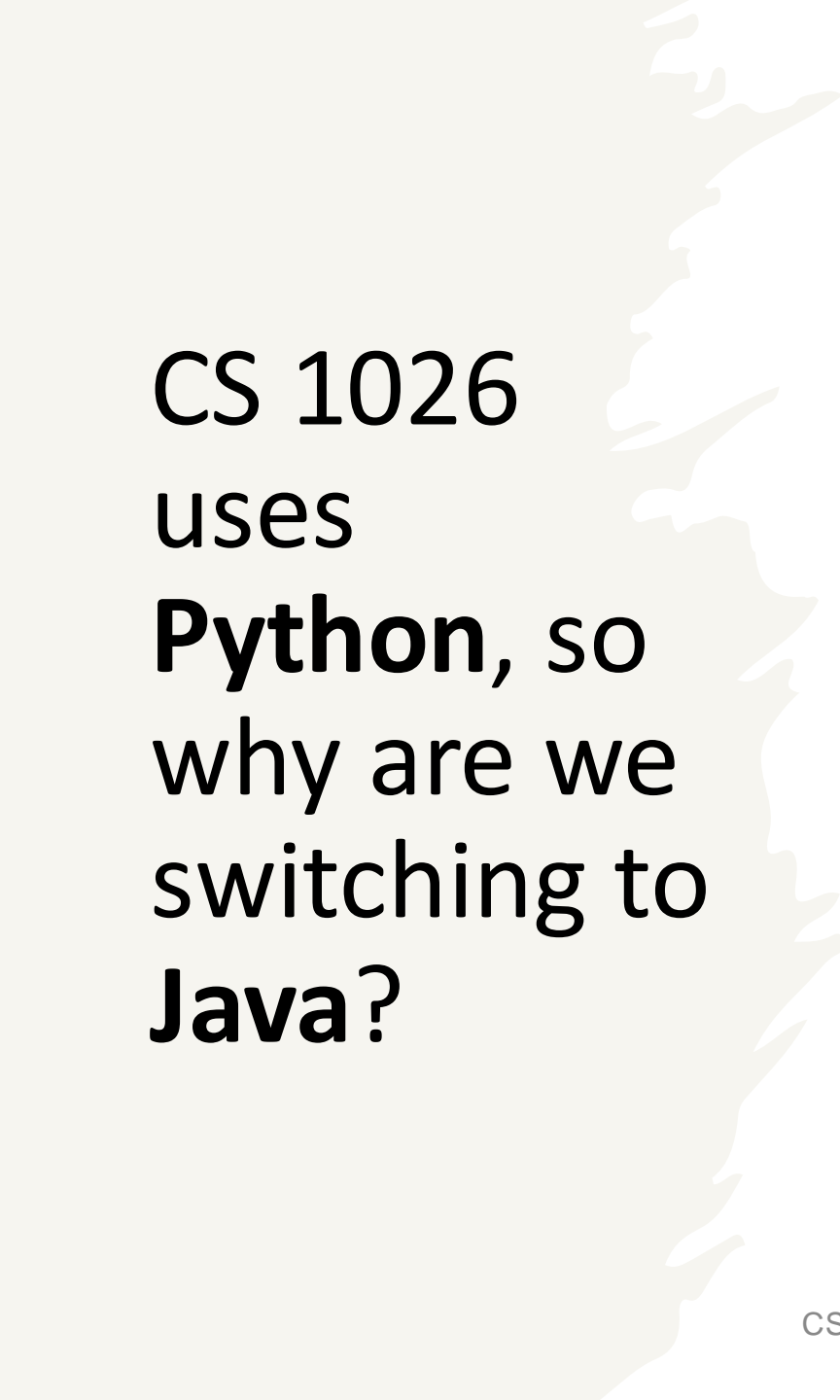
- Python allows us to teach programming concepts without confusing students with complex syntax.

Python:

```
print("Hello World")
```

Java:

```
public class HelloWorld {  
    public static void main (String[] args)  
    { System.out.println("Hello World");  
    }  
}
```



CS 1026 uses **Python**, so why are we switching to **Java**?

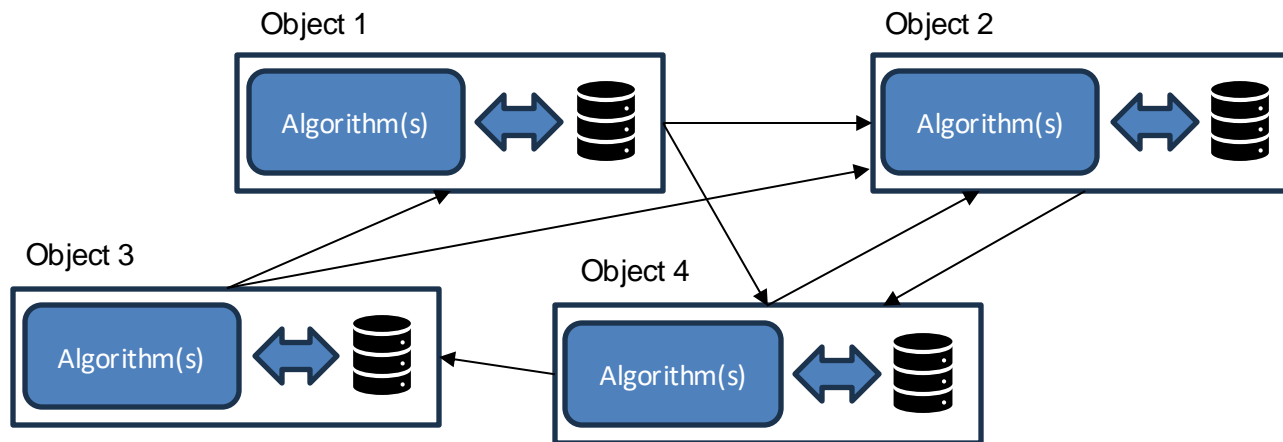
- Four out of the five most commonly used programming languages are **statically typed** (Python is **dynamically typed**)
- Java provides more flexibility than Python.
- For example, lists are ***automatically expanded*** in Python, so we cannot specify a particular manner for enlarging or shrinking a list.
- Java programs generally **run faster** than Python programs.

Course Topics

- Algorithms and data structures
- Object-oriented design concepts:
encapsulation, abstraction, inheritance,
polymorphism
- Exceptions and debugging
- Arrays and linked structures
- Abstract data types and their
implementations: stacks, queues, lists,
trees
- Recursion and memory management
- Sorting algorithms

Recall: Object Oriented Design (OOD) Principles

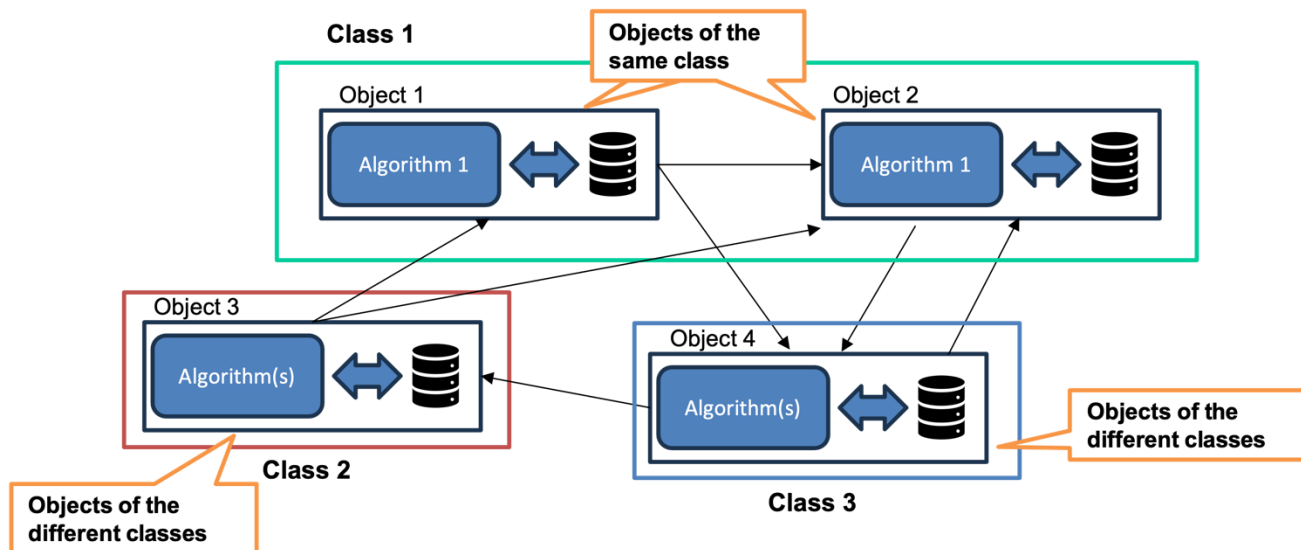
- OO design principles, such as **Encapsulation**, **Abstraction**, **Inheritance**, and **Polymorphism**, aim to simplify the task of designing correct programs.



Object-Oriented Design

Encapsulation

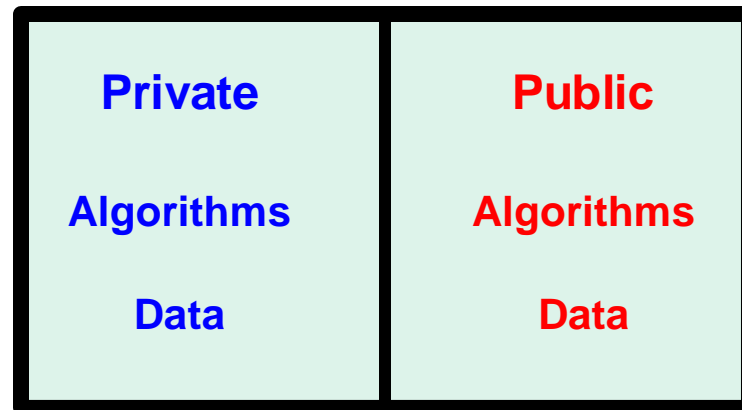
- The algorithms and data of an object SHOULD NOT be modifiable by objects of other classes.
- This allows us to design classes **independently from each other**.



Abstraction

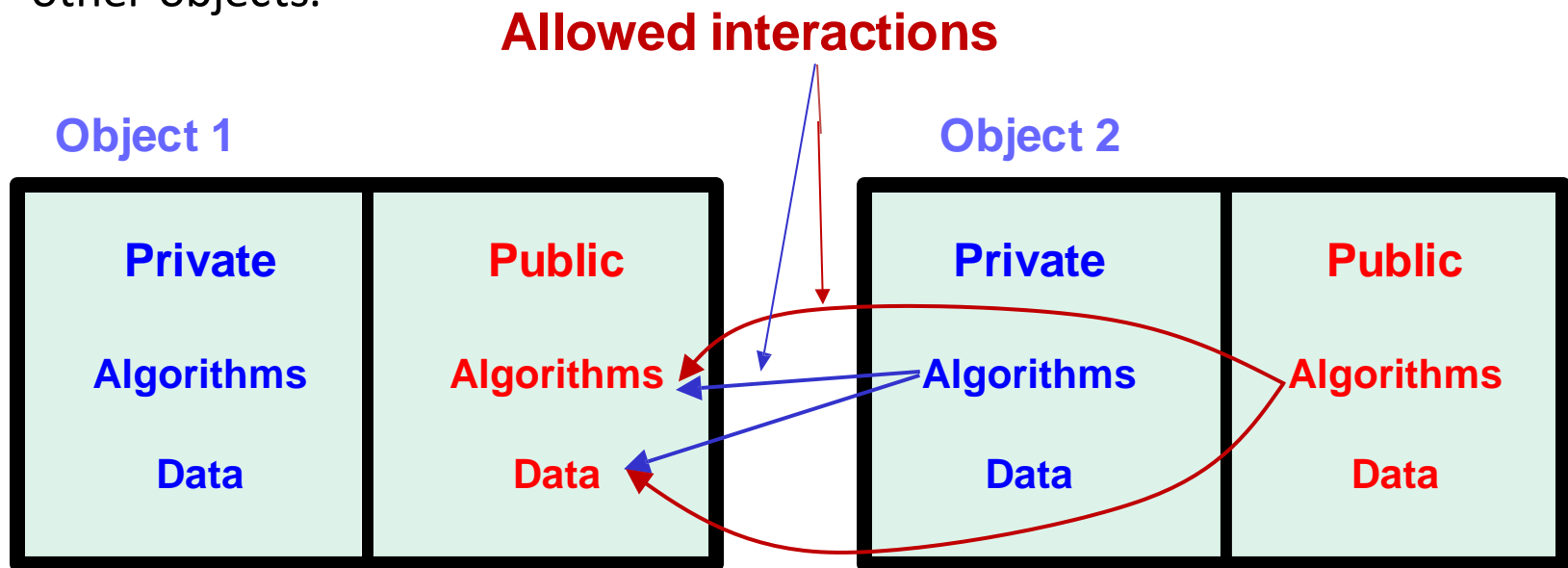
- An object has algorithms and data of two types:
 - **Private** algorithms and data are not accessible to other classes.
 - **Public** algorithms and data are accessible to other classes.

Object



Abstraction (cont.)

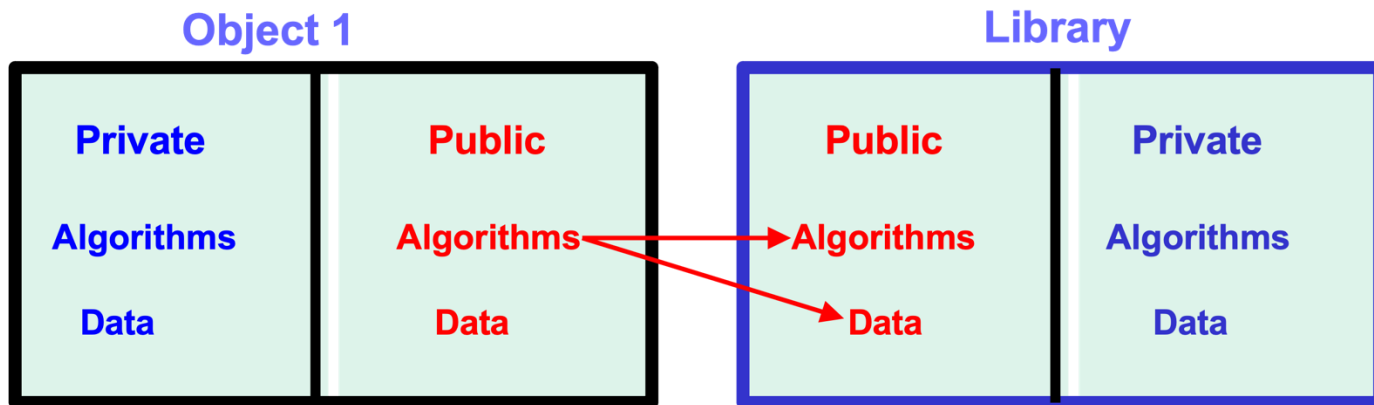
- Public algorithms and data enable abstraction, allowing objects to interact without needing to understand the internal implementation of other objects.



Abstraction (cont.)

Abstraction is a powerful design principle that allows:

- decomposing a program into smaller, simpler parts
- code reuse
- library use
- simpler code modification



Polymorphism

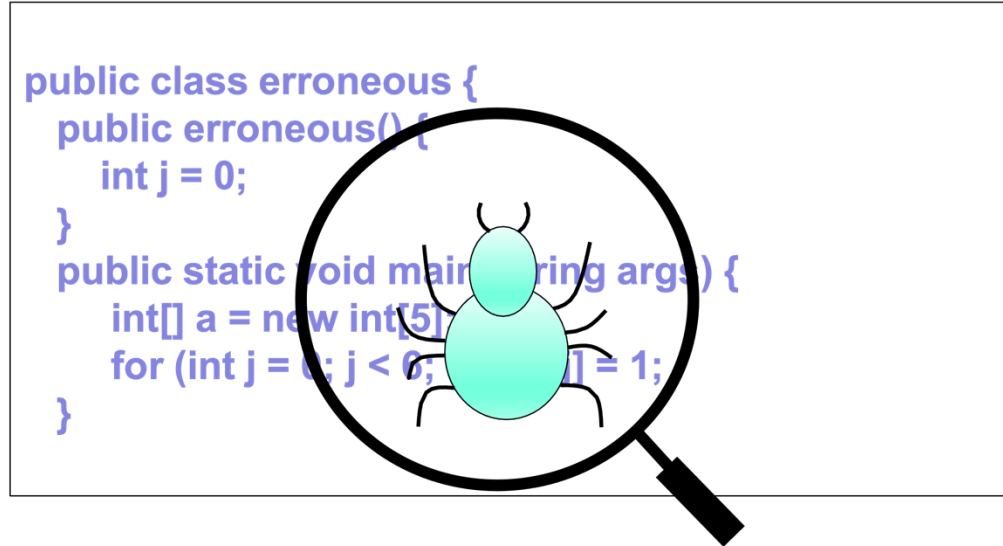
- Polymorphism is a key concept in object-oriented programming that enables us to apply one algorithm in different ways. It helps us create more flexible and reusable code.
- Think of it this way: Consider an 'algorithm' that can be applied to different tasks, such as sorting numbers, organizing text, or arranging files. Despite the differences in tasks, they all follow the same 'algorithm' in their unique way.

Course Topics

- Algorithms and data structures
- Object-oriented design concepts:
encapsulation, abstraction, inheritance,
polymorphism
- Exceptions and debugging
- Arrays and linked structures
- Abstract data types and their
implementations: stacks, queues, lists,
trees
- Recursion and memory management
- Sorting algorithms

Exceptions and Debugging

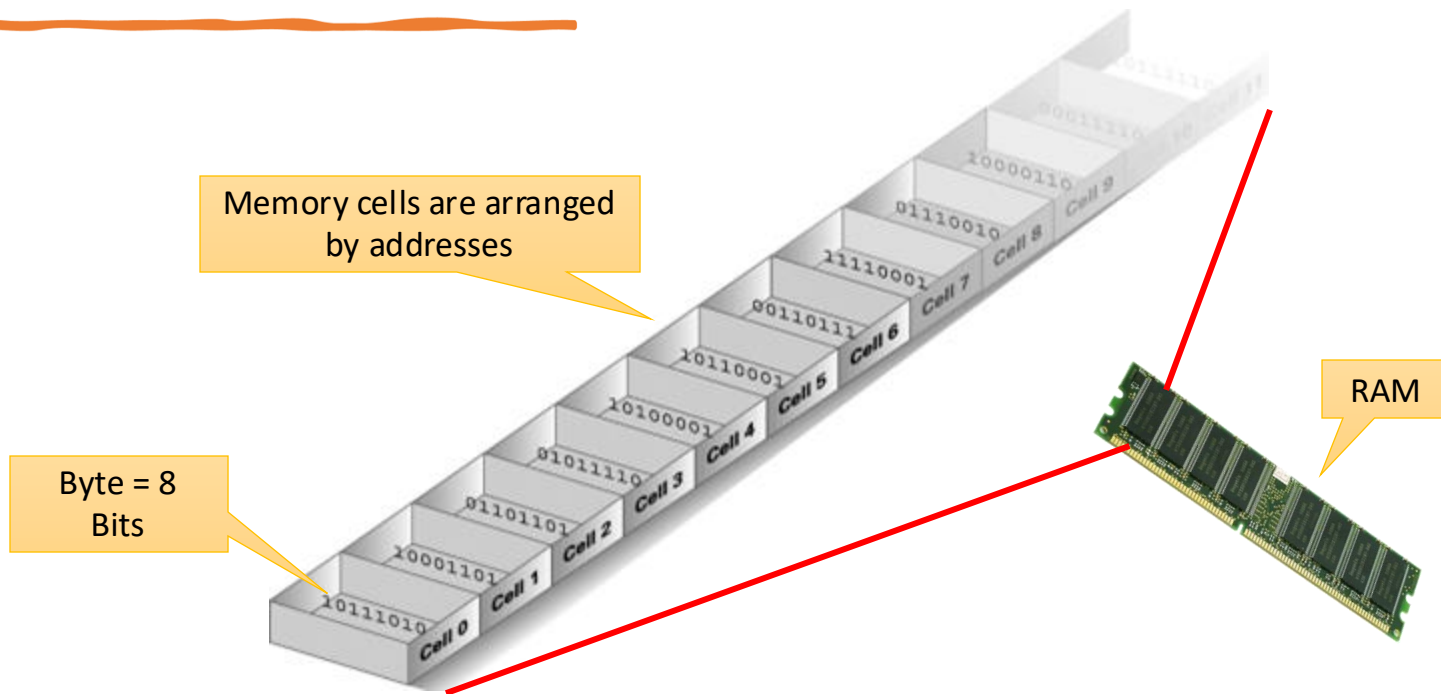
- **Exceptions:** Java mechanism to deal with errors detected during the execution of a program.
- **Debugging** is the process of finding and correcting errors in a program.



Course Topics

- Algorithms and data structures
- Object-oriented design concepts:
encapsulation, abstraction, inheritance,
polymorphism
- Exceptions and debugging
- Arrays and linked structures
- Abstract data types and their
implementations: stacks, queues, lists,
trees
- Recursion and memory management
- Sorting algorithms

Memory Cells



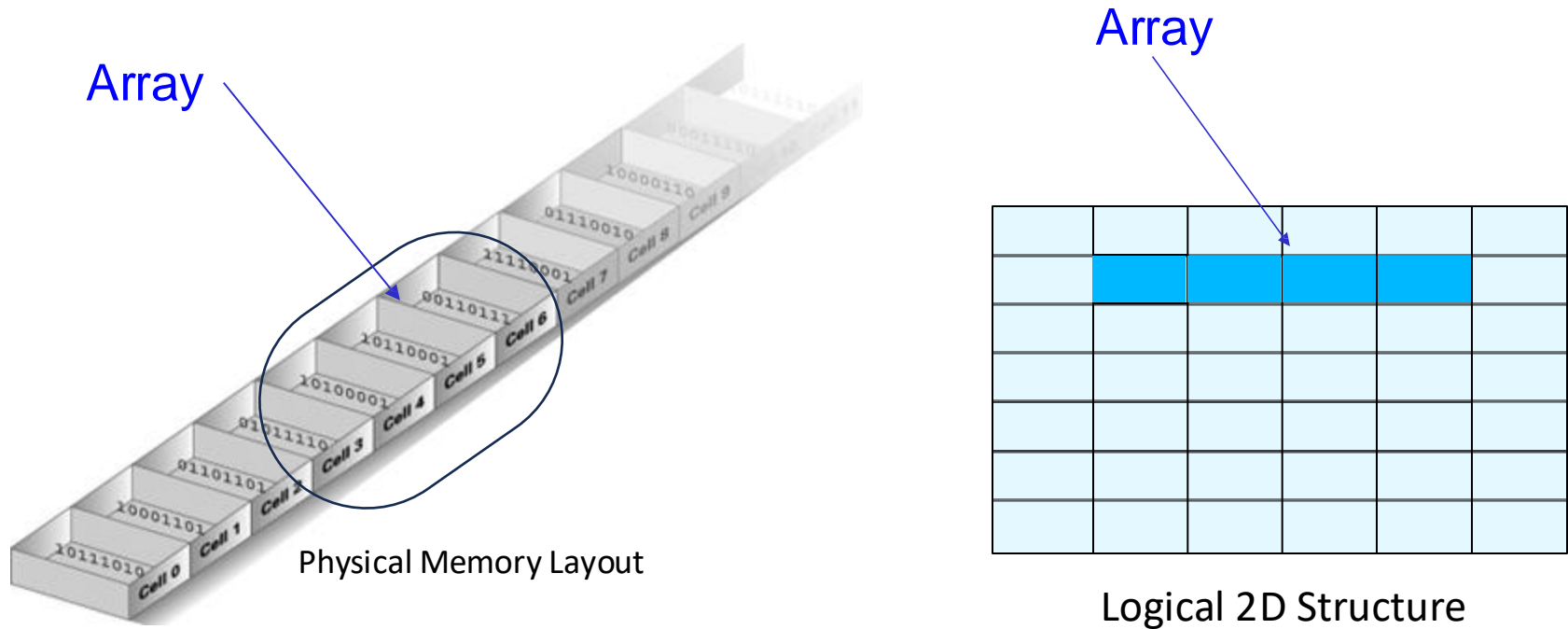
- The memory of a computer consists of a set of memory cells, each having a **unique address**.

Memory Cells Addresses

- **Address:** uniquely identifies one cell in the computer's main memory
 - **Addresses** are **unique** numbers.
 - These numbers are assigned consecutively, starting at zero.
 - Numbering the cells in this manner associates an order with the memory cells.



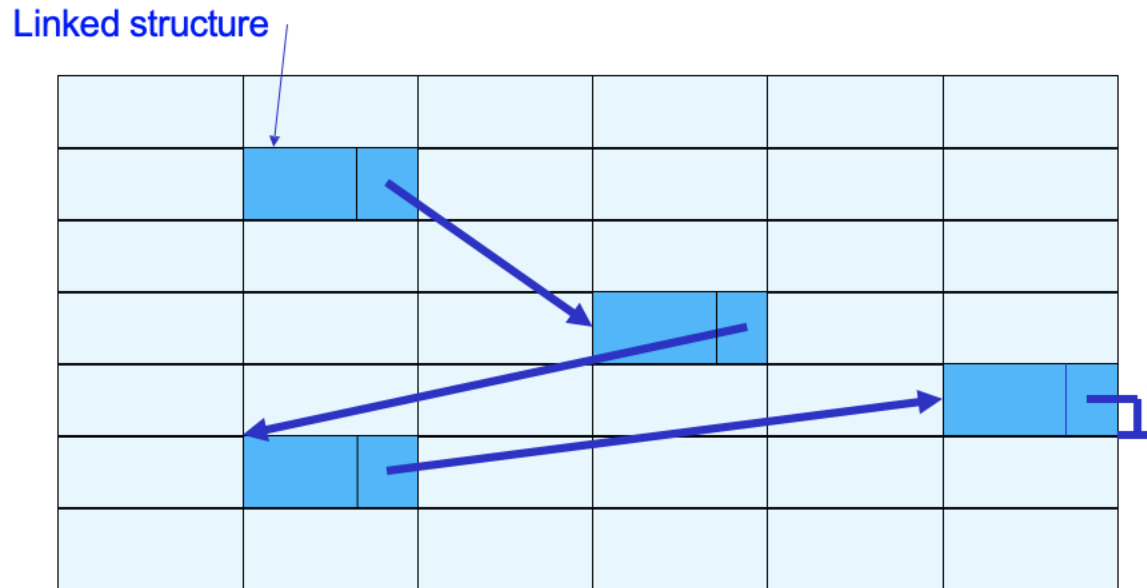
Arrays and Linked Structures



- An array consists of a set of **adjacent** memory cells.

Arrays and Linked Structures

- A linked structure consists of a set of cells that might be located **anywhere** in memory.



Memory Logical 2D Structure

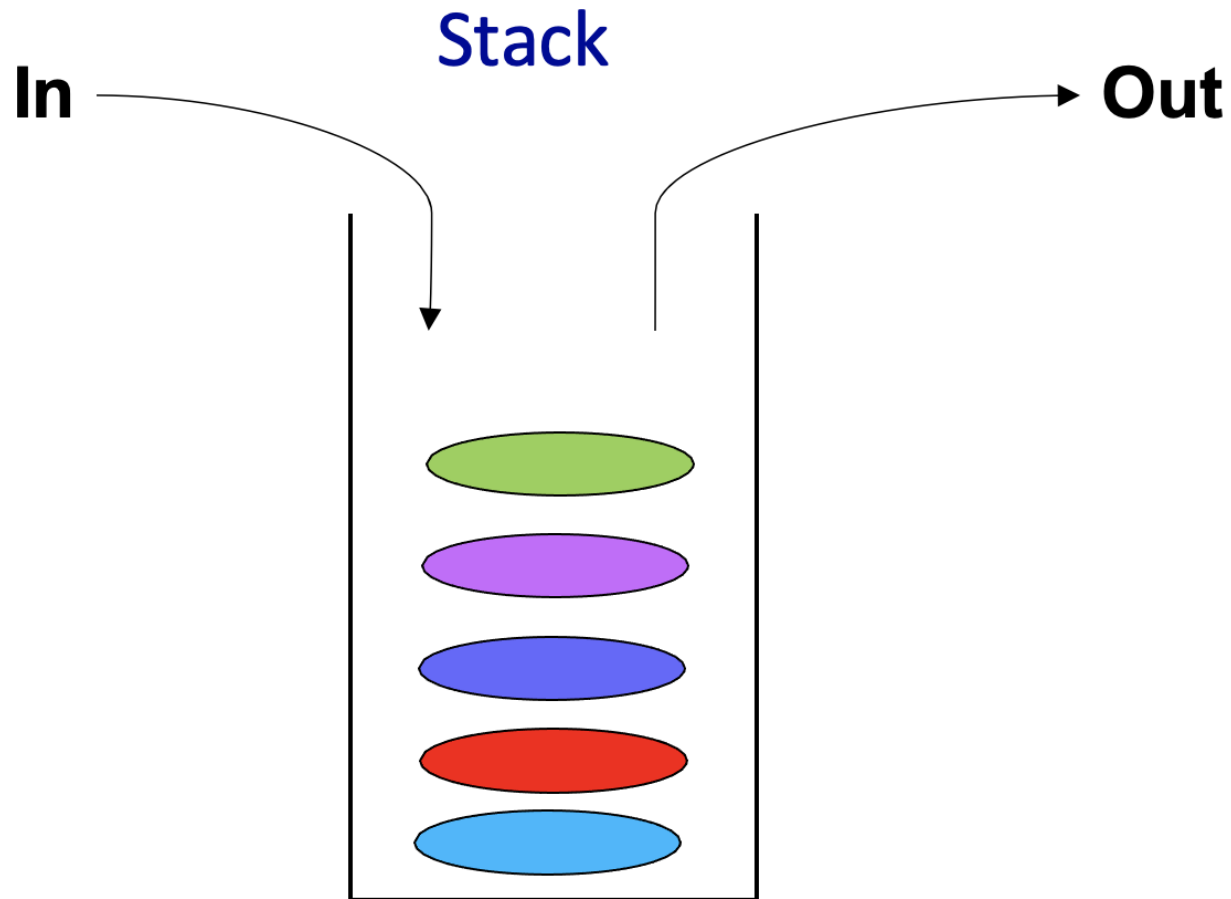
Course Topics

- Algorithms and data structures
- Object-oriented design concepts:
encapsulation, abstraction, inheritance,
polymorphism
- Exceptions and debugging
- Arrays and linked structures
- Abstract data types and their
implementations: stacks, queues, lists,
trees
- Recursion and memory management
- Sorting algorithms

Abstract Data Types (ADTs)

An Abstract Data Type (ADT) is a **conceptual model** or **blueprint** for a data structure. It is defined by the operations that can be performed on it rather than ***how*** it is implemented.

ADTs focus on ***what*** the data structure can do, not ***how*** it does it.

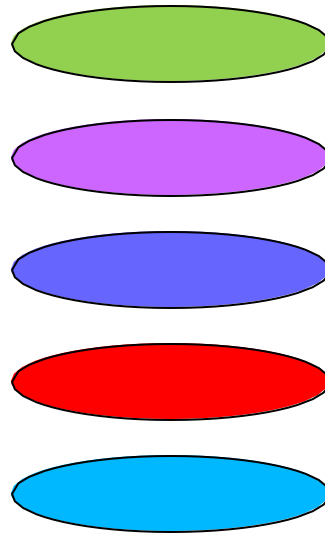


- A stack is open at one end

Queue

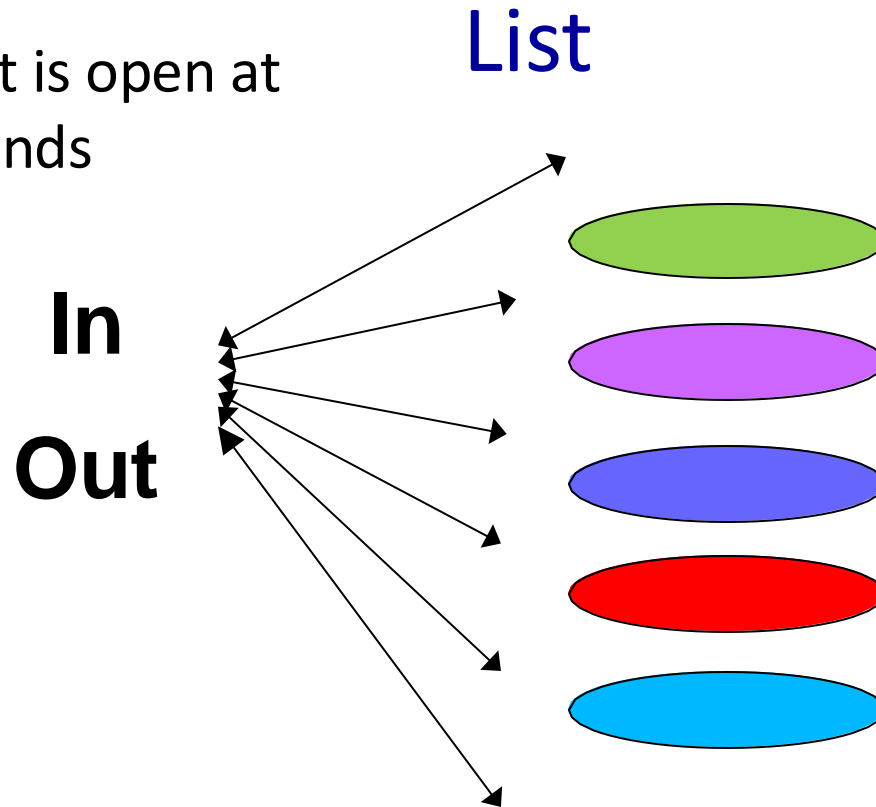
In

- A queue is open at two ends



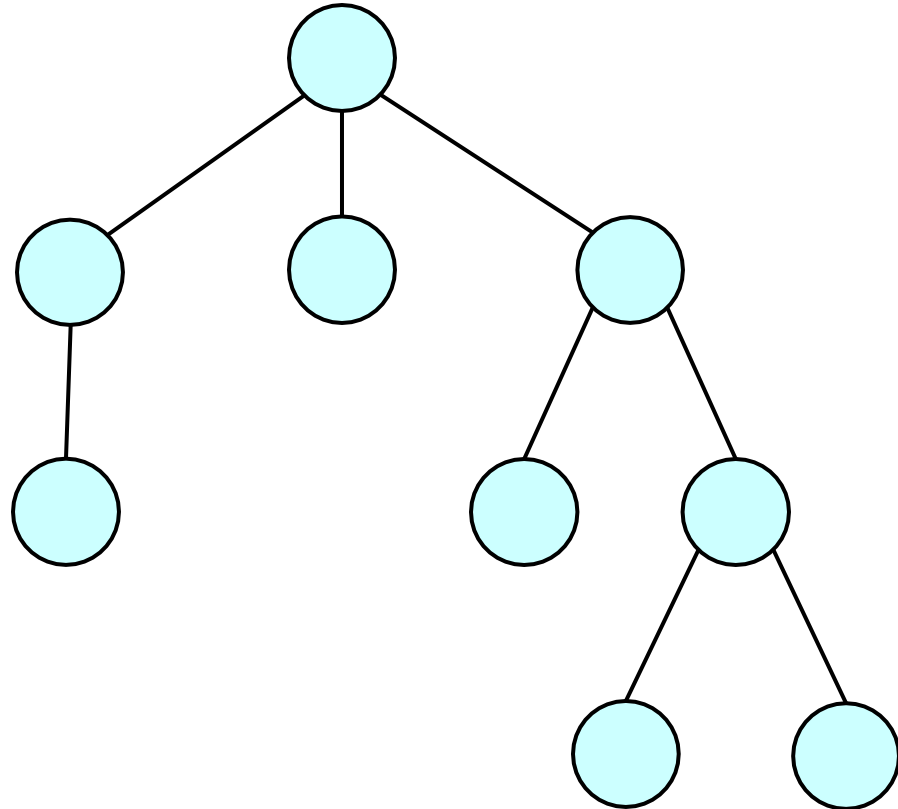
Out

- A list is open at all ends



Tree

- A tree is not a linear data structure. It is a hierarchical data structure

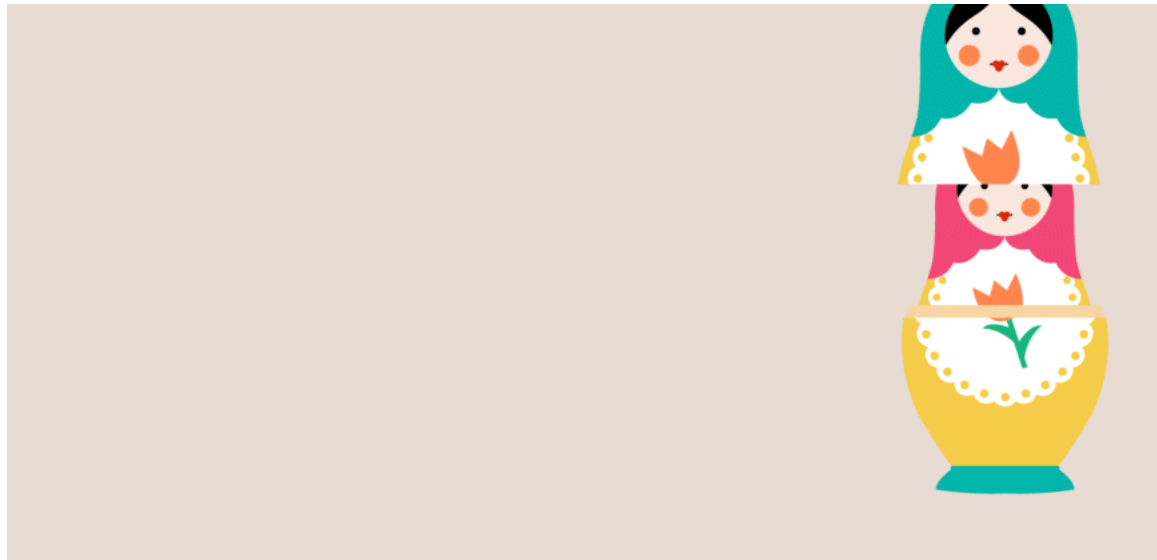


Course Topics

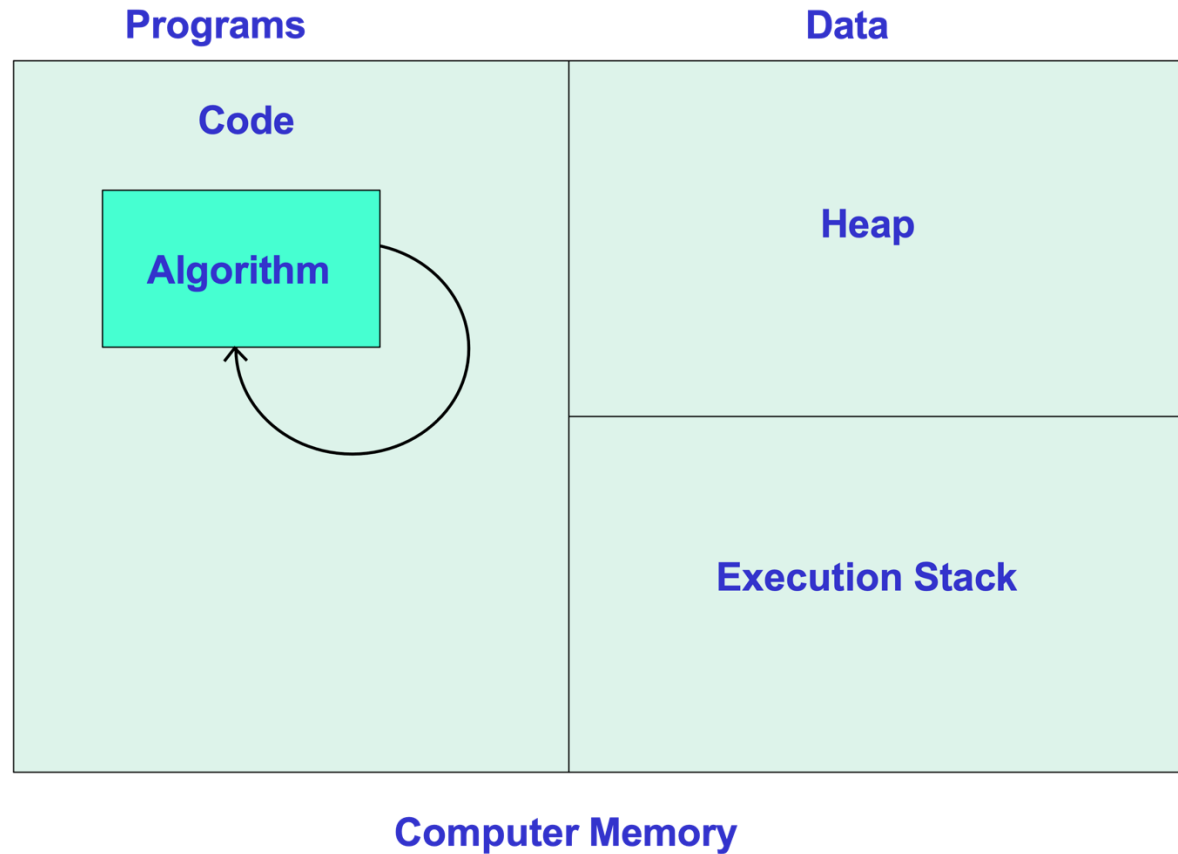
- Algorithms and data structures
- Object-oriented design concepts:
encapsulation, abstraction, inheritance,
polymorphism
- Exceptions and debugging
- Arrays and linked structures
- Abstract data types and their
implementations: stacks, queues, lists,
trees
- Recursion and memory management
- Sorting algorithms

Recursion

- You keep breaking things down until they become simple, then solve the small problem.
- After solving the small problem, you work your way back up and solve the bigger problem.



Recursion and Memory Management





Thank you