TRINITY COLLEGE
DEPARTMENT OF COMPUTER SCIENCE

CPSC 215: Data Structures and Algorithms
Instructor: Dr. Chandranil Chakraborttii
Spring 2025

Assignment 1

Academic Honesty Policy

In working on programming assignments, you may discuss broad issues of interpretation and understanding and general approaches to a solution. However, conversion to a specific solution or to program code must be your own work. Programming assignments are expected to be the work of the individual student, designed, and coded by him or her alone. Violations are easy to identify and will be dealt with promptly according to the Academic Integrity and Intellectual Dishonesty outlined in the Student Handbook.

- Copying another person's programs or encouraging or assisting another person to commit plagiarism is cheating. In particular, the following activities are strictly prohibited:
- Giving and receiving help in the actual development of code or writing of an assignment.
- Looking at another person's code or showing your code to another person.
- Sharing a copy of all or part of your code regardless of whether that copy is on paper or in a computer file.
- Turning in the work of any other person(s) (former students, friends, textbook authors, people on the Internet, etc.) and representing it as your own work.
- Fabricating compilation or execution results.
- Use of AI ChatBots for any help regarding the assignment is **strictly prohibited.**

The penalty for cheating is a failing grade (F) for the course, and the student is asked to appear at an Academic Dishonesty Hearing. In addition, the College also places a record of the incident in the student's permanent record. It is your responsibility to protect your work from unauthorized access. Do not discard copies of your programs in public places. Do not leave computers unattended and copies of output lying around.

Objectives

- To understand and implement array-based data structures using generics.
- To design and develop code that demonstrates basic encapsulation, abstraction, and polymorphism.

Array Implementation Using Generics

Part 1: Getting Started

- Open Eclipse and create a Java Project (File -> New -> Java Project)
- Give a project name (For example: assignment_1_CPSC_215)
- Go to Project -> Right Click -> New Package
- Give a package name (For example: assignment_1_CPSC215)
- Now to create the classes, go to package -> Right Click -> New Java class.
- Paste contents of the given files. Keep the first line in place (package "package_name";)
- Understand the code structure before starting to write code.

You are provided the following classes.

- FixedArray.java (Need to complete)
- FixedArrayTest.java (Don' change)

Part 2: FixedArray Class

Review the FixedArray.java class that demonstrates the use of generics. The file includes the following features:

- **Encapsulation**: The data members array and size are private, ensuring they are accessed and modified only through methods.
- **Abstraction**: Methods like insert, remove, swap, display, and clear provide a clear interface for interacting with the array.
- **Polymorphism**: The use of generics allows the array to store any data type (e.g., String, Integer, Double).

Tasks:

1. **Complete the Following Methods in FixedArray.java**:
   - insert(T element): Adds an element to the array at the end. Ensure that no elements can be added when the array is full.
   - remove(int index): Removes an element from a specific index and shifts elements appropriately.
   - clear(): Clears all elements in the array and resets the size to zero.
   - swap(int index1, int index2): Swaps two elements in the array based on their indices.

2. **Provided Methods**:
   - display(): Displays all elements in the array.
   - getSize(): Returns the current size of the array.

Part 3: Main Class

Review Main.java contains pre-written code to test the FixedArray class. It demonstrates the application of object-oriented principles:

- **Encapsulation**: The FixedArray class is accessed through its public methods.
- **Polymorphism**: The generic array stores different data types such as String and Number.

Tasks (contd):

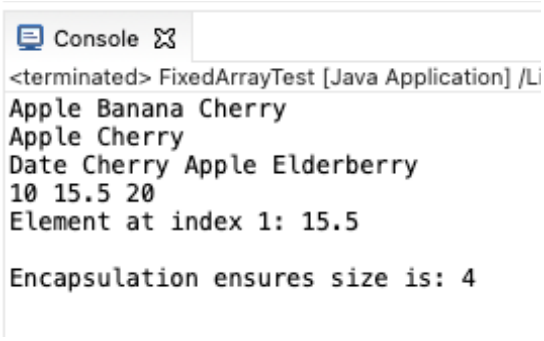
3. **Run the provided code to:**
   - Test the insert method by adding elements to the array.
   - Test the remove method by deleting an element from a specific index.
   - Test the swap method by swapping two elements in the array.
4. **Review the use** of abstraction and encapsulation by analyzing how data is accessed and manipulated through methods.
5. **Experiment by adding additional test cases** to verify edge scenarios (e.g., attempting to insert into a full array, removing an invalid index).

Note:

When completing the methods, you need to handle the following edge cases:

- Attempting to insert an element into a full array.
- Trying to remove an element at an invalid index (negative index or index out of bounds).
- Swapping elements when one or both indices are invalid.
- Ensuring no NullPointerException occurs when displaying or clearing the array.

Sample Output:



Part 4: Submission

Upon the completion of your lab, upload your submission to the Moodle course website as a single Zip file. (FixedArray.java and FixedArrayTest.java)
