## Department of Computing

## CS 212: Object Oriented Programming

## Lab 07: Polymorphism & Inferances

## Date: 1-04-2024

## Time: 9:00pm- 12:00 pm

## Instructor: Mr. Jaudat Mamoon

## Lab Engineer: Engr. Masabah Bint E Islam

## Task# 1

Develop a simple Inventory Management System (IMS) that can add, update, remove, and list inventory items. This task will help students understand object-oriented programming concepts, collections, and file handling in Java.

Create an Item class with properties such as id, name, price, and quantity. Include a constructor, getters, and setters for these properties. Implement an InventoryManager class with the following functionalities:

* **Add Item:** Add a new item to the inventory.
* **Update Item:** Update the details of an existing item.
* **Remove Item:** Remove an item from the inventory based on its id.
* **List Items:** Display a list of all items in the inventory.

Create a simple text-based user interface in the main class that allows users to perform the above operations. Use the console for input and output.

## Task-2:

Several requirements are presented below. Use these requirements to write all the necessary classes and/or interfaces for a solution to the problem. You may implement your solution as you wish, but you will be graded on the appropriateness of your solution to the requirements. You may use whatever constructors or additional methods you wish where necessary.

Define a class that can represent Animals. Animals have two behaviors; they can speak() and they can move(). By default, when an animal moves, the text “This animal moves forward” is displayed. By default, when an animal speaks, the text “This animal speaks” is displayed. A general Animal should not be able to be instantiated.

Define also two classes, Goose and Lynx that are Animals. Both Goose and Lynx behave such that where “animal” is displayed in speak () or move (), ”goose” or ”lynx” is displayed by the appropriate classes.

Finally, any instance of Goose can fly (), just as any Flying object can. An Airplane is also a Flying object. Define the Airplane class such that it is Flying and make sure that any instance of Goose is also Flying. The specific behaviors when instances of either class fly () are up to you. Instances of either Goose or Airplane should be able to be stored in a variable of type Flying.

Write a test application that demonstrates all the capabilities of the classes / objects including references of both Animal and Flying.

## Task# 3:

