**COSC 436: Object-Oriented Design and Programming**

**In-class Exercise: Adapter Design Pattern 1**

**Tasks: understand Adapter design pattern.**

**Note: You are encouraged to work with your neighbors. This exercise does NOT need to use computers.**

Background: You are running an animal show in Las Vegas. Your performance team includes several mallard ducks. They can **fly** and **quack**. Below is the code that represents your show.

**// Duck interface defines the**

**// behavior of ducks**

**public** **interface** Duck {

**public** **void** quack();

**public** **void** fly();

}

**// Mallard Duck you hire in your team**

**public** **class** MallardDuck **implements** Duck {

**public** **void** quack() {

System.***out***.println("Quack");

}

**public** **void** fly() {

System.***out***.println("I'm flying");

}

}

**// Client code that makes the duck perform**

**public** **class** ShowTime {

**public** **static** **void** main(String[] args) {

MallardDuck duck = **new** MallardDuck();

*performance*(duck);

}

**// This is the Client interface**

**static** **void** performance(Duck duck) {

duck.quack();

duck.fly();

}

}

=====================================================================================================================

New situation: One day, your ducks eat something bad and can’t perform. You find some turkeys to help you perform, but they do **NOT** **quack**, and they **fly** shorter distance. Below is the code for Turkey.

**// Defines behavior of Turkey**

**public** **class** Turkey {

**public** **void** gobble() {

System.***out***.println("Gobble gobble");

}

**public** **void** fly() {

System.***out***.println("I'm flying a short distance");

}

}

=====================================================================================================================

Problem: We can’t change *performance(Duck)* method. We can’t make Turkey *quack* or fly longer… Please find out a solution that can make Turkey join your performance.

**Think about the Adapter design pattern. Discuss with your neighbors.**

**After 5 minutes, check the back of this page for a partial solution. =======================🡺**

**Below is a partial solution, please fill your code on the lines to complete the implementation.**

// This is the **Adapter**

**public** **class** TurkeyAdapter **implements** Duck {

// wrap a Turkey inside

Turkey turkey;

// constructor

**public** TurkeyAdapter(Turkey turkey) {

**this**.turkey = turkey;

}

// let’s make Turkey quack

**public** **void** quack() {

// PLEASE FILL YOUR CODE HERE

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

}

// let’s make Turkey fly longer

**public** **void** fly() {

// PLEASE FILL YOUR CODE HERE

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

}

}

// The new ShowTime with Turkey

**public** **class** ShowTime {

**public** **static** **void** main(String[] args) {

Turkey turkey = **new** Turkey(); // Hire a turkey to help

turkey.gobble(); // Just want to try if the turkey is alive

turkey.fly(); // Just want to try if the turkey is alive

// let’s create an adapter

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

// let’s perform

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

}

// Note that this client interface is **NOT** changed

**static** **void** performance(Duck duck) {

duck.quack();

duck.fly();

}

}

1. **Please write down the output after running the new code.**
2. **Can you draw a class diagram for the new structure of using Turkey in the show?**