Lecture 13 Design Patterns Clicker Questions – Solutions

1.

```
public class Pianist extends Musician {
    private Piano _piano;
    public Pianist (Piano p) {
        _piano = p;
    }
}
```

Which design pattern(s) does this code use?

A. Inheritance
B. Composition
C. Both
D. Neither

ANSWER: C

The Pianist class uses both the inheritance and composition design patterns. The Pianist inherits from the Musician class, and thus a Pianist "is a" Musician. The Pianist is also composed of the Piano class. Thus, both patterns are used.

2.

```
public class BrownClass {
    private Student[] _students;
    //code elided
    public Student getStudent(int index) {
        if(index < 0 || index > _students.length) {
            System.out.println("you are trying to access an incorrect
index");
            return null;
        }
        return _students[index];
    }
}

public class CS15Class extends BrownClass {
    private CS15Student[] _cs15Students;
    //code elided
    @Override
    public Student getStudent(int index) {
        return _cs15students[index];
    }
}
```

What would happen if the following code is called?

```
CS15Class cs15 = new CS15Class();
cs15.getStudent(-1);
```

A. A CS15Student is returned
B. An array out of bounds error is thrown
C. Null is returned
D. A Student is returned

ANSWER: B

If we call getStudent() with a negative index value, we will get an array index out of bounds error because the index is less than zero, and is not contained within the limits of the array indices.

3.

```
public class Cupcake {
    //constructor and other methods elided
    public void addChocolate() {
        this.setFlavor(Constants.CHOCOLATE);
        this.setSugarPerServing(this.getSugarPerServing()
                    + Constants.CHOCO_SUGAR_SERVING);
    }
}
```

This codes a superclass Cupcake for potential subclasses. You want to create ChocolateCupcake and VanillaCupcake subclasses. Should this code be changed?

A. No – superclass contains all identical functionality for each of its subclasses
B. Yes – superclass should have an addVanilla() method
C. Yes – superclass shouldn't have addChocolate() since only one subclass uses it
D. Yes – superclass should keep addChocolate() since you might make more chocolate flavored subclasses.

ANSWER: C
Superclasses should implement functionality that all subclasses will need. In this case, addChocolate() is a method for only one subclass, and it isn't necessary within the superclass.

4.
What is the relationship between a MacBook and Laptop?

A. Inheritance: a Laptop "is a" MacBook
B. Inheritance: a MacBook "is a" Laptop
C. Composition: a MacBook does what a Laptop does
D. Composition: a Laptop does what a MacBook does

ANSWER: B

If we were to design these classes, we would use the inheritance design pattern since a MacBook “is a” Laptop. A MacBook is not composed of Laptops, and a Laptop is not composed of MacBooks. Finally, not all Laptops are MacBooks. Thus, the correct inheritance relationship is B – a MacBook is a Laptop and would extend a Laptop superclass.