CSE 122 Section Homework - Section 14 Solutions

This assignment focuses on interfaces.

1. The following code represents two coffee shops, `SeattleSunshine` and `Seafab`, each with methods to serve different types of coffee.
    a. Identify and correct any issues in the code so it will compile and run as intended. There are two issues. Rewrite an entire *line* of code if it is missing or incorrect.

    Hint: the errors are *not* in the `main` method.

```
import java.util.*;
public class Coffee {
    public static void main(String[] args) {
       CoffeeShop sunshineCoffee = new SeattleSunshine();
       CoffeeShop fabCoffee = new Seafab();
       System.out.println(sunshineCoffee.serveLatte());
       System.out.println(fabCoffee.serveLatte());
       System.out.println(sunshineCoffee.serveMocha());
       System.out.println(fabCoffee.serveMocha());
    }
}

public class SeattleSunshine implements CoffeeShop {
    public String serveLatte() {
        return "Here is your latte from Seattle Sunshine!";
    }

    public String serveMocha() {
        return "Here is your mocha from Seattle Sunshine!";
    }
}

public class Seafab implements CoffeeShop {
    public String serveLatte() {
        return "Here is your latte from Seafab!";
    }

    public String serveMocha() {
        return "Here is your mocha from Seafab!";
    }
}

public interface CoffeeShop {
    public String serveLatte();
    public String serveMocha();
}
```

First rewritten line: `public class SeattleSunshine implements CoffeeShop {`

Second rewritten line: `public String serveMocha();`

b. Circling back to the hint, since we know the main method is correct, then these two lines of code are also correct:

```
CoffeeShop sunshineCoffee = new SeattleSunshine();
CoffeeShop fabCoffee = new Seafab();
```

Would the debugged code run if these lines were changed to:

```
SeattleSunshine sunshineCoffee = new SeattleSunshine();
Seafab fabCoffee = new Seafab();
```

Why or why not?

The code **would** run if these lines were changed. This is because `sunshineCoffee` is specifically typed as `SeattleSunshine`. Similarly, `fabCoffee` is typed as `Seafab`. Since both directly reference their classes, the `CoffeeShop` interface does not have to be specified here. The setup allows direct access to instance methods in each of the classes without relying on the interface.

c. Would the debugged code run if these lines were changed to:

```
SeattleSunshine sunshineCoffee = new CoffeeShop();
Seafab fabCoffee = new CoffeeShop();
```

Why or why not?

The code **would not** run if these lines were changed because `CoffeeShop` is an interface, and interfaces cannot be directly *instantiated*. Only classes that implement the interface can be instantiated.

Reminder: to instantiate means to create an object or an instance of a class.

2. Why is it important to avoid using implementation details in comments?

Comments are meant to provide an explanation about a class or method to help others, maybe even people who do not know Java, understand the purpose of the code.

3. Suppose I have the classes `Aquarium`, `Seal`, `Squid`, and `Otter` and the interface `Animal`. If the `Animal` interface had the methods `eat()`, `sleep()`, and `makeFriends()`, each of which return a String, write the interface and include a class comment.

```
1    // This is the Animal interface. This interface
2    // described eating, sleeping, and get animal type
3    // behaviors for any animals, such as making friends.
4    public interface Animal {
5       public String eat();
6       public String sleep();
7       public String makeFriends();
8    }
```