Part A: SubstringChecker 4 points

+1/2 class SubstringChecker implements Checker

+1/2 declare private instance variable of type String

+1 constructor
   +1/2 SubstringChecker(String goalString)
   +1/2 initialize instance variable to parameter

+2 accept method
   +1/2 public boolean accept(String text)
   +1 1/2 determine whether to accept
      +1/2 attempt to find instance variable in text
         (either call indexOf, contains, or compare with substrings)
   +1 return correct boolean value in all cases

Part B: AndChecker 4 points

+1/2 class AndChecker implements Checker

+1/2 declare private instance variable(s) capable of storing two Checker objects

+1 constructor
   +1/2 AndChecker(Checker c1, Checker c2)
   +1/2 initialize instance variable(s) to parameters

+2 accept method
   +1/2 public boolean accept(String text)
   +1 1/2 determine whether to accept
      +1/2 attempt to call accept(text) on both stored Checkers
   +1 return correct boolean value in all cases

Part C: yummyChecker 1 point

+1 correctly assign yummyChecker
PART A:

public class SubstringChecker implements Checker {
    private String goalString;

    public SubstringChecker(String goal)
    {
        goalString = goal;
    }

    public boolean accept(String text)
    {
        return (text.indexOf(goalString) != -1);
    }
}

PART B:

public class AndChecker implements Checker {
    private Checker checker1;
    private Checker checker2;

    public AndChecker(Checker chk1, Checker chk2)
    {
        checker1 = chk1;
        checker2 = chk2;
    }

    public boolean accept(String text)
    {
        return checker1.accept(text) && checker2.accept(text);
    }
}

PART C:

yummyChecker = new AndChecker(new NotChecker(aChecker),
                                new NotChecker(kChecker));
Write the `SubstringChecker` class that implements the `Checker` interface. The constructor should take a single `String` parameter that represents the particular substring to be matched.

```java
public class SubstringChecker implements Checker {
    private String sub;
    public SubstringChecker(String s) {
        sub = s;
    }
    public boolean accept(String text) {
        return text.indexOf(sub) >= 0;
    }
}
```

Part (b) begins on page 18.
Write the `AndChecker` class that implements the `Checker` interface. The constructor should take two `Checker` parameters.

```java
public class AndChecker implements Checker {
    private Checker uno;
    private Checker dos;
    public AndChecker(Checker one, Checker two) {
        uno = one;
        dos = two;
    }

    public boolean accept(String text) {
        return uno.accept(text) && dos.accept(text);
    }
}
```

Part (c) begins on page 20.
In writing your solution, you may use any of the classes specified for this problem. Assume that these
classes work as specified, regardless of what you wrote in parts (a) and (b). You may assume that the
declarations for aChecker, kChecker, and yummyChecker in the code segment above have
already been executed.

Write your /* code to construct and assign to yummyChecker */ below.
Checker nart = new NotChecker(new SubstringChecker("artichokes"));
Checker nkal = new NotChecker(new SubstringChecker("kale"));
Checker yummyChecker = new AndChecker(nart, nkal);
Write the `SubstringChecker` class that implements the `Checker` interface. The constructor should take a single `String` parameter that represents the particular substring to be matched.

```java
public class SubstringChecker implements Checker
{
    String substring;

    public SubstringChecker(String s)
    {
        substring = s;
    }

    public boolean accept(String text)
    {
        if (text.indexOf(substring) != -1)
            return true;
        return false;
    }
}
```

Part (b) begins on page 18.
Write the `AndChecker` class that implements the `Checker` interface. The constructor should take two `Checker` parameters.

```java
public class AndChecker implements Checker {

    Checker checkerOne;
    Checker checkerTwo;

    public AndChecker(Checker o, Checker t) {
        checkerOne = o;
        checkerTwo = t;
    }

    public boolean accept(String text) {
        return checkerOne.accept(text) && checkerTwo.accept(text);
    }
}
```

Part (c) begins on page 20.
In writing your solution, you may use any of the classes specified for this problem. Assume that these classes work as specified, regardless of what you wrote in parts (a) and (b). You may assume that the declarations for aChecker, kChecker, and yummyChecker in the code segment above have already been executed.

Write your /* code to construct and assign to yummyChecker */ below.

```
yummyChecker = new NotChecker(ArdChecker(aChecker, kChecker));
```
Write the SubstringChecker class that implements the Checker interface. The constructor should take a single String parameter that represents the particular substring to be matched.

```java
public class SubstringChecker
{
    private String MyChecker;

    public SubstringChecker (string text)
    {
        MyChecker = text;
    }
}
```

Part (b) begins on page 18.
Write the AndChecker class that implements the Checker interface. The constructor should take two Checker parameters.

```java
public class AndChecker {
    private Checker myCheckerOne;
    private Checker myCheckerTwo;
    public AndChecker(Checker firstChecker, Checker secondChecker) {
        myCheckerOne = firstChecker;
        myCheckerTwo = secondChecker;
    }
}
```

Part (c) begins on page 20.
In writing your solution, you may use any of the classes specified for this problem. Assume that these classes work as specified, regardless of what you wrote in parts (a) and (b). You may assume that the declarations for aChecker, kChecker, and yummyChecker in the code segment above have already been executed.

Write your /* code to construct and assign to yummyChecker */ below.
Overview

This question focused on inheritance, class design, and Boolean logic. Students were provided with the Checker interface that contains a single boolean method named accept. In part (a) they were required to design and implement the SubstringChecker class (which implements the Checker interface) so that the accept method returns true if its string parameter contains a specific substring. This involved selecting an appropriate instance variable, defining a constructor that takes a String as a parameter, and implementing the accept method using appropriate String methods. In part (b) students were required to implement a different class that implements Checker, the AndChecker class. This also involved selecting appropriate instance variables, defining a constructor that takes two Checkers as parameters, and implementing the accept method so that it calls the accept method on both Checkers and returns the AND of the two results. In part (c) they were required to complete the construction of a Checker object that computed a particular Boolean function.

Sample: A4a
Score: 9

In part (a) the student provides a correct class header and a correct declaration of a private instance variable. The constructor is completely correct; it initializes the instance variable to the parameter. The accept method also is completely correct. It finds the index of the instance variable string in the parameter string and returns the correct result. Note that it correctly returns true when the returned index is 0. The student earned 4 points for this part.

In part (b) the student provides a correct class header and correct declarations of the private instance variables. The constructor is completely correct; it initializes the instance variable to the parameter. The accept method also is completely correct. It calls the accept method correctly on each of the instance variables and returns the results combined with &&. The student earned 4 points for this part.

In part (c) the student provides a correct instantiation of the required Checker object and assigns it to the correct variable. Note that the student re-implements the SubstringChecker objects for “artichokes” and “kale,” which is allowed. The student earned 1 point for this part.

Sample: A4b
Score: 7

In part (a) the student provides a correct class header but lost ½ point for not declaring the instance variable as private. The constructor is completely correct; it initializes the instance variable to the parameter. The accept method also is completely correct. It finds the index of the instance variable string in the parameter string and returns the correct result. The student earned 3½ points for this part.

In part (b) the student provides a correct class header but lost ½ point for not declaring the instance variables as private. The constructor is completely correct; it initializes the instance variable to the parameter. The accept method also is completely correct. It calls the accept method correctly on each of the instance variables and returns the results combined with &&. The student earned 3½ points for this part.

In part (c) the student provides an incorrect instantiation of the required Checker object by creating a NotChecker object from an AndChecker object rather than an AndChecker object of two NotChecker objects. This was a commonly seen error. The student earned no points for this part.
Sample: A4c
Score: 3

In part (a) the student provides an incorrect class header and lost ½ point. A correct declaration of a private instance variable is provided. The constructor is completely correct; it initializes the instance variable to the parameter. The student does not provide an accept method and lost 2 points. The student earned 1½ points for this part.

In part (b) the student provides an incorrect class header and lost ½ point. Correct declarations of the private instance variables are provided. The constructor is completely correct; it initializes the instance variables to the parameters. The student does not provide an accept method and lost 2 points. The student earned 1½ points for this part.

In part (c) the student does not provide any code, so no points were earned.