# Question 1 – Coding Style Guidelines

# Question 2 – Comment Reader

## Input to CommentReader

/\*

\* Comment on a class

\*/

public class TestClass {

private int i1; // A first number

private double d2; // Another number

/\* this explain a bit \*/ private int i3; /\* explain more \*/ // Another comment

/\*\*

\* **@param** d

\*/

public TestClass(double d)

{

i1 = 1;

d2 = d;

i3 = 1;

}

// Will this be fine /\*

public int sum()

{

// Returns a value

return i1 + i3;

}

/\* and another

\* multi

\* line /\* will this break?

\*/

## Output of CommentReader

\* Comment on a class

A first number

Another number

this explain a bit explain more Another comment

\*

\* @param d

Will this be fine /\*

Returns a value

and another

\* multi

\* line /\* will this break?

# Question 3 – Ball Container

## Different implementations for getVolume()

Our first implementation to getVolume() was:

public double getVolume() {

double sum = 0;

for (Ball ball : balls)

{

sum += ball.getVolume();

}

return sum;

}

Our second implementation to getVolume() was holding a variable in the class and updating it on every add and remove, and zeroing the variable when clearing the container.

The first implementation is better if the calls to getVolume() are infrequent. As there is no extra work to be done on update and remove, on the rare occasions we call getVolume(), iterating over the ArrayList won't matter so much. Also this implementation uses less memory than holding an extra variable.

However, if the calls to getVolume() are frequent, we don't want to waste time constantly iterating over the ArrayList, so using an updated variable is more efficient.

## Change to @requires for add()

### Needed alterations to the specification

The addition of the @requires doesn't need changes to the specification, as nowhere in the specification referenced what happens if the parameter ball was null.

### Needed alterations

The addition of the @requires doesn't need changes to the implementation. We can now forgo the check if ball is null but we don't have to.

### Strength of specification after alteration

The specification is now weaker, as the @requires is longer and the @effects remains the same.

# Question 4 – Multiple choice

## Choice between two specifications

The correct answer is א, it cannot be established which specification is stronger. For a specification to be stronger than a specification we require that (1) the @requires be stronger and (2) the @effects be more specific. In the question has a stronger @requires but has a stronger @effects, so we cannot establish a stronger specification.

## Specification of the method find()

### implements

The correct answer is ד, in this case also implements and . It implements because they have the same effect, and it implements because if the color is the list it will return the location of the color. It does not implement on account of the error thrown if the color is not found.

### implements

The correct answer is ו, the implementation only implements . The implementation may rely on the fact that the list is ordered in order to find the location. For the list may be not ordered, and therefore the implementation may return the wrong value. For , the specification throws an error if not found, therefore doesn't implement it. Finally for , in the same way as the function may return the wrong value as the list may be not ordered.

### implements

The correct answer is ג, the implementation also implements . For , if color is in the list the location will be returned which is the only return value specified. For and , if the color isn't in the list, the implementation will throw an error rather than return -1, so doesn't implement these two specifications.

### implements

The correct answer is ו, the implementation only implements . The specification doesn’t specify what happens if color isn't in the list, therefore the implementation may return the value which is not as specified in all the other specifications.