Questions for Objective-C and Storage Management for Android versus iOS app’s

1. Does the Objective-C/Swift runtime for MacOSX applications (as opposed to iOS applications) use garbage collection? Explain the different data control approaches used by Java and Objective-C/Swift – garbage collection versus automatic reference counting (ARC). Consider ease of use for the app developer, and efficiency from both a time and space perspective.

Discuss the differences between the Java approach and the Objective-C/Swift iOS approach in the context of the following code construction (which is not syntax specific to either language):

```java
String myString = "1-The quick brown fox jumped over the lazy dog";
for (int i = 0; i < 10000; i++){
    myString = new String("I am new string \(i\)");
}
```

Explain the difference in managing the 10,000 new strings that would be created by the body of the loop (all of which but the last are not used). Explain the difference in how these strings are reclaimed in Objective-C/Swift (ARC) as compared to Java’s garbage collection.

2. Suppose I define a Java instance method with the following signature:

```java
public String getFormattedString(String base, String formatSpecifier);
```

Give the signature for a corresponding Objective-C method. Give a sample call to the Objective-C method.

3. Consider the Objective-C messages (method calls) below. Assuming the same user-defined objects and class names exist in a Java program, show the corresponding Java statements for:

```objective-c
anObject = [myObject1 method1: @"I am string"];  
anObject = [myObject1 appendTo: @"I am " theString: @"anotherString"];  
anotherObject = [[myObject1 method1:@"I am string"] valueAt: myIndex encoding: yourEncoding];
```

4. Properties. When creating a property in objective-C you can specify a modifier to determine how values are assigned to the instance variable in the setter operation. The modifier can be one of: copy, assign (weak), or retain (strong). The body of the setter gets created automatically in the implementation of the class at the point of the @synthesize directive. The synthesize is inserted into the implementation by default if its not explicitly placed. Indicate for each of the code templates below whether it’s a template for a setter method that gets created by the synthesize for copy, assign (weak), or retain (strong) storage class.

```objective-c
- (void) setStr: (NSString*) inStr {
    if (self->str != inStr){
        [self->str release];
        self->str = [inStr retain];
    }
}
- (void) setAge: (int) inAge {
    self->age = inAge;
}
```