

Java/C++ : Check Your Understanding

Name: _____

1. Equality

In Java, what is the difference between the comparison operator, `==`, and the comparison method, `equals`, when comparing objects?

2. Object References

Consider the following two innocent looking methods:

```
StringBuffer getText ()  
{  
    return myText;  
}
```

```
String getName ()  
{  
    return myName;  
}
```

Because every object variable in Java is a reference, explain why the first method has the potential to break the encapsulation of the object, while the second method does not. Why is this not as much of a problem in C++?

3. In order to properly store a class within an STL container, what constructors, methods, and operators must your class implement? When are these methods automatically written for you? What additional functions must be written to make your class comparable within the STL?

4. What is the main difference between running programs in C++ and Java? What impact does that have in terms of compile-time versus run-time errors?

5. Header files

Header files *bar.h* and *foo.h* are shown below on the left and middle, respectively. A program *main.cpp* is shown on the right.

```

#ifndef _BAR_H           #ifndef _FOO_H           #include <iostream>
#define _BAR_H          #define _FOO_H
int globalvalue;        int globalvalue;        #include "bar.h"
                                                                    #include "foo.h"

class Bar               class Foo               int main()
{                       {
  int myValue;          int myValue;          {
                                                                    return 0;
                                                                    }
};                       };
#endif                  #endif

```

1. The file *main.cpp* (which includes both *bar.h* and *foo.h*) is compiled, as are the files *bar.cpp* and *foo.cpp*. All the *.o* files are linked to create an executable.

```

g++ -g -Wall -c foo.cpp -o foo.o
g++ -g -Wall -c bar.cpp -o bar.o
g++ -g -Wall -c main.cpp -o main.o
ERROR in file included from main.cpp:3: bar.h:4: redefinition of `int globalvalue' foo.h:4: `int globalvalue' previously declared her

```

Explain why the `#ifndef` directives at the top of the files *foo.h* and *bar.h* do not stop the redefinition problem.

2. Suppose that the code in *main.cpp* is changed so that no use of the class *Foo* occurs in *main.cpp*. However, the files *foo.o*, *bar.o*, and *main.o* are still linked to create an executable. The error message from the compilation changes as shown below.

```

g++ -g -Wall -c foo.cpp -o foo.o
g++ -g -Wall -c bar.cpp -o bar.o
g++ -g -Wall -c main.cpp -o main.o
g++ -g -Wall -o main foo.o bar.o main.o
ld: fatal: symbol `globalvalue' is multiply defined: (file foo.o and file bar.o);
ld: fatal: symbol `globalvalue' is multiply defined: (file foo.o and file main.o);
ld: fatal: File processing errors.

```

Why does this error occur, and is it different from the previous error?