QUEEN MARY, UNIVERSITY OF LONDON
DCS/115 INTRODUCTION TO PROGRAMMING

Class Test Monday 20th November 2000 10.05-10.55

Please fill in your name here: **ANSWERS**

College registration number here:

Department email here: mmh@dcs.qmw.ac.uk Course Code (e.g G500) here:

All answers to this test should be written on the test sheet, but you may use spare paper for rough working, or the space on the back page if you cannot fit your answers into the space provided. Answer as many of the questions as you can. This is an "open book" test: you are allowed to access your notes and textbook.

1) Explain in the spaces given what is meant by:

a) "Compiler" with regards to Java

A compiler is normally a program that converts a high-level language to the machine code obeyed directly by a computer. The Java compiler, however, converts Java to an intermediate form called "Java byte code" which is obeyed by running another program, the Java byte code interpreter.

b) "Aliasing" caused by assignment in Java

Aliasing means that two or more variables refer to the same object. In Java if two variables `a` and `b` are of an object type (i.e. any type except a primitive type) the assignment `a=b` causes `a` and `b` to refer to the same object, that is they are aliases of each other.

c) "Short-circuit" with the `&&` operator

The `&&` operator in Java is a "short-circuit" operator. This means that in an expression `a&&b` where `a` and `b` are any boolean expressions, `b` is only evaluated if `a` evaluates to `true` (since if `a` evaluates to `false` we know the whole expression evaluates to `false` without evaluating `b`). This will be noticeable if the evaluation of `b` has a side effect.

d) "Scope" with regard to variables in Java

The scope of a variable declared in a particular declaration is the portion of the code of a program in which the use of that variable's name refers to the variable in the declaration. For example, a variable local to a method has scope up to the `}` matching the first `{` before its declaration.

2) In each of the cases below, write what would be printed when the fragment of Java code is executed (assume we have the declaration `int a, b, c`):

a)
```
a=22;
b=10;
c=3;
System.out.println(a-b/c);
```
19

b)
```
a=7;
b=2;
b*=a+1;
System.out.println(b);
```
16

c)
```
a=5;
b=20;
c=b-a-3;
System.out.println(++c);
```
13

d)
```
a=2;
b=5;
c=a>b?b*a:b-a;
System.out.println(c);
```
3

e)
```
a=9;
b=a--;
c=a;
a=b*2;
System.out.println(a+" "+b+" "+c);
```
18 9 8

f)
```
a=8;
b=3;
double d = (double)a/b;
c=(int)d*2;
System.out.println(c);
```
4

3) Suppose we have three `int` variables, `a`, `b`, and `c`, which have been given values.

a) Write a fragment of code that will print `Yes` if `b` has a value that is in between `a` and `c` (but not equal to either) and `No` otherwise.

```
if(b>a && b<c || b>c && b<a)
    System.out.println("Yes");
else
    System.out.println("No");
```

b) Suppose we have a method with signature `boolean hit(int force)` and a variable `punchbag` on which that method can be called. Write a fragment of Java code which calls `hit` on `punchbag` with argument `b`, until either the method returns `false` or it has been called `c` times. The variable `a` may be used to store temporary `int` values if needed.

```
a=0;
while(a<c && punchbag.hit(b))
  a++;
```

4) A `Machine` object has the following public methods:

```
int getTemp()
void addFuel(int amount)
Grommet pushButton()
```

A `Grommet` object has a public method:

```
int getQuality()
```

A `Machine` object can be constructed with a zero argument constructor.

This represents a situation in which a machine produces grommets simulated by calling the `pushButton` method. The `addFuel` method simulates adding an amount of fuel to a machine, and the `getTemp` method simulates checking its temperature. The `getQuality` method simulates testing the quality of a grommet.

a) Write a fragment of Java code which prompts the user to enter the amount of fuel to be added, and then simulates starting a machine and adding that amount of fuel to it.

```
BufferedReader in =
    new BufferedReader(new InputStreamReader(System.in));
System.out.print("Enter amount of fuel to be added: ");
int amount = Integer.parseInt(in.readLine());
Machine m = new Machine();
m.addFuel(amount);
```

b) Write a fragment of code that simulates starting a machine and producing grommets until its temperature drops below 20, then prints how many grommets have been produced.

```
int count = 0;
Machine m = new Machine();
while(m.getTemp()>=20)
  {
    Grommet g = m.pushButton();
    count++;
  }
System.out.println(count+" grommets produced");
```

c) Suppose we have a variable `g` of type `Grommet` that refers to some `Grommet` object, and a variable `m` of type `Machine` that refers to a `Machine` object. Write a fragment of Java code that simulates `m` producing 100 grommets and prints how many of them were of a higher quality than the grommet represented by `g`.

```
int count=0, qcount=0;
while(count<100)
  {
    count++;
    Grommet g1 = m.pushButton();
     if(g1.getQuality()>g.getQuality())
        qcount++;
  }
System.out.println(qcount+" high quality grommets produced");
```

Use the space below if you need any additional space to answer any of the questions.