1	20/
2	80/
3	20/
4	40/
5	10/
6	15/
7	15/
	200/

CMSC 331 Second Midterm Exam

Name: _____ Student ID#:_____

You will have seventy-five (75) minutes to complete this closed book exam. Use the backs of these pages if you need more room for your answers. Describe any assumptions you make in solving a problem. We reserve the right to assign partial credit, and to deduct pts for answers that are needlessly wordy.

0. Warm up (0 pts)

You have 3000 dates and a camel that can carry at most 1000 dates at a time. The camel eats a date for every meter it moves. What's the most number of dates you can move over 1000 meters? 532

1a. A silly question (10 pts)

Given the definition of the Silly class in the box to the right, what gets printed when you execute these statements in a method in some other class?

```
class Silly {
    int number = 5;
    void bump(Silly s, int num) {
        s.number++;
        num++;
    }
}
```

```
Silly s = new Silly();
Silly t = new Silly();
int n = 8;
s.bump(t, n);
System.out.println(s.number + " " + t.number + " " + n);
```

5 6 8

1b. An average question (10 pts)

Define a class `Average` whose instance objects keep a running average of the last two doubles they were asked to "ingest". The class should have two methods: `value` and `ingest`. The parameterless `value` method returns the average of the last two numbers ingested. The `ingest` method takes a double and returns the average of it and the previous double ingested. Make as much of the class private as possible. If you make any assumptions, be sure they are reasonable and document them in your code.

```
public class Average {
   private double first, second;
   public Average() { first = second = 0.0; }
   public double value() { return (first + second)/2.0;}
   public double ingest (double x) {
            first = second;
             second = x;
             return value();}
}
```

3. True/False (80 pts)

For each of the following questions, circle T (true) or F (false).

T F A char is just a String of length one.
T F An abstract class cannot be instantiated.
T F Methods declared in an interface are automatically public.
T F Constants may be declared in an interface.
T F The size of an array is part of its type.
T F In BlueJ, a striped box means the class is not compiled.
T F A RuntimeException need not be caught.
T F You can attach more than one Listener to a Component.
T F You can attach a Listener to more than one Component.
T F In an if test, zero counts as false, other integers as true.
T F You can do integer arithmetic with a char variable.
T F The break statement can only be used within a loop.
T F The continue statement can only be used within a loop.
T F Applets can only be run from within a browser.
T F To create an applet, you must subclass the Applet class.
T F A Java file can legally contain more than one public class.
T F Anonymous inner classes can access the fields of the enclosing class.
T F Every method must have at least one return statement.
T F Java arrays are restricted to primitive types
T F If a method has no parameters, you must still call it with parentheses after the method name.
T F Any class containing an abstract method must be an abstract class
T F A class can extend a regular class but it must implement an abstract class
T F If a class is defined to be final, it can not be instantiated.
T F A wrapper object is an object that corresponds to one of Java's primitive types.
T F A static method defined for a class C can not be called on an instance of C.
T F In the model-view-controller design pattern, the model implements the user interface.
T F While JAVA's AWT is bigger and slower that Swing, it if usually considered to be more flexible and to yield better looking interfaces.
T F A Java class can only have one constructor.
T F Every Java Class must either define or inherit at least one destructor method.
T F A method must declare a return type or void if no value is returned
T F In a method, the keyword `this` refers to the current object and the keyword `that` refers to the class to which the object belongs.
T F The first version of Java was named "Pine" after a tree outside the designer's office.
T F Only one class can be defined in a file.
T F If all of a class's fields are private and it has no setter methods, its objects will necessarily be immutable.
T F Java does cooperative scheduling instead of pre-emptive scheduling
T F Every thread has its own private run-time stack
T F A try statement can have any number of associated catch statements, including none.
T F Java represents exceptions, but not errors, as objects.
T F A private method can only be accessed from other methods within the same class.
T F Javadoc is java's debugger

3. What Happens? (20 pts)

First, step through the given program, updating the values of all the variables, until you get to the `System.out.println` statement. *Then* fill in the following table, telling (1) the simplest possible way to print the variable, and (2) what value will be printed for it. If a variable cannot be accessed, tell why.

```
public class Major {
    int a = 1, b = 2, c = 3;
    Extra extra = new Extra();
    public static void main(String args[]) {
        int b = 4, d = 5;
        Minor minor = new Minor(b);
        minor.printAll();
    }
}
class Minor extends Major {
    int c = 6, e = 7;
    static int f = 8;
    Minor(int param) { b = e = param; }
    void printAll() {
        System.out.println(expression from
table);
    }
}
class Extra {
    int f = 9;
    Extra() { f = 10; }
    class Inner { int g = 11; }
    Inner inner = new Inner();
}
```

Variable	Simplest expression to access variable	Value
a in Major	a	1
b in Major	b	4
b in main	Impossible – `Can't access local variable outside function`	
c in Major	super.c	3
c in Minor	c	6
d in main	Impossible – `Can't access local variable outside function`	
e in Minor	e	4
f in Minor	f	8
f in Extra	extra.f	10
g in Inner	Extra.inner.g	11

4. Example code (40 pts)

```
1  interface Inter {
2      int number();
3  }
4  abstract class Abs {
5      static int foo = 12;
6      int number() { return 5; }
7      abstract int ace();
8  }
9  final class Sub extends Super {
10      Sub(int bar) { foo = bar; }
11      public int number() { return 10; }
12      int ace() { return 13; }
13      int dub(int i) { return 2 * i; }
14  }
15  public class Super extends Abs implements Inter {
16      public int number() { return 11; }
17      public static void main(String args[]) {
18          Super s1 = new Super();
19          Super s2 = new Sub(16);
20          System.out.println(      );
21      }
22      int twice(int x) { return 2 * x; }
23      public int thrice(int x) { return 3 * x; }
24      int ace() { return 1; }
25      String dub(String s) { return s + s; }
26  }
```

Answer these questions for the code in the box. If more than one answer is possible, just give one.

(a) (5 pts) Does `Super` correctly implement `Inter`? Why or why not?

Yes, because it defines `number()`

(b) (5 pts) In line 11, it is a syntax error to omit the word `public`. Why?

Because `number()` in `Sub` overrides `number()` in `Super()`, and you can't make an overridden function less public.

(c) (4 pt) The method defined on line _____ overrides the method defined on line _____.

24 and 12, *or* 11 and 16, *or* 16 and 6 (any one pair)

(d) (3 pt) The method defined on line _____ overloads the method defined on line _____.

13 and 25 (or 25 and 13)

(e) (3 pt) To print out `foo` with its original value of `12`, you could put a print statement between lines _____ and _____.

17 and 19 (must be in a method, and must be done before an instance of `Sub` is created in line 19)

(f) (5 pts) Which of the above classes and/or interfaces have default constructors?

`Super` (only)

(g) (3 pt) What effect does the keyword `final` on line 9 have?

Prevents any other class from extending this one

(h) (12 pts) Give the value for each expression in this table or "error" if an error would occur.

`s1.ace()` 1	`s2.foo` 16	`s1.dub(6)` error	`s2.number()` 10
`s2.ace()` 13	`s1.twice(3)` 6	`s2.dub(7)` error	`s1.dub("8")` 88
`S1.foo` 16	`s2.twice(3)` 6	`s1.number()` 11	`s2.dub("9")` 99

5. A simple method (10 pts)

(a) Write a method `divisible` that takes two integer parameters and returns true if its first parameter is divisible by its second parameter and false otherwise. For full credit, keep your method body as short as possible.

```
boolean divisible(int x, int y) {
          return x % y == 0;}
```

(b) Should your `divisible` method be declared `static`? Why or why not?
Yes, because it has nothing to do with any particular object.

6. Classes, interfaces and abstract classes (15 pts)

A freshman who lives down the hall reveres you as a master programmer and seeks your sage advice on many things. One day she asked "Wise one, I am confused by new types of Java classes I saw in class today. Since I was young I have known only the ordinary Java class but now I have learned of the *Interface* and the *Abstract Class*. When should I use each of those rather than an ordinary class or the other of those strange new classes. You replied "Ah, Young Grasshopper ...

Use either an interface or an abstract class when you do not want another programmer (or you yourself) to create instances of them, but only to subclass them. Use an interface if you want the java compiler to **require** that subclass provide implementations of all of the methods in the interface. Use an abstract class if you want to provide some of the definitions. Use an interface if you want the new subclass to extend yet a third class.

7. An exceptional question (15 pts)

(a) (5 pts) Briefly describe the conceptual differences between exceptions and errors.

Errors are considered to be bugs in the logic of your program and should be fixed. Exceptions are usually caused by external objects not being as expected -- e.g., a file you are asked to read does not exist or is read protected. Exceptions need to be checked for and either reported to the user or recovered from, if possible. It is impossible to "fix" exceptions so that they will not occur.

(b) (5 pts) Java's mechanisms for handling errors and exceptions are similar. Briefly describe how they are alike and how they differ.

Exception and Errors are both subclasses of the Throwable class. The Exception class is used for exceptional conditions that user program should catch. With exception class we can subclass to create our own custom exception. Error defines exceptions that are not excepted to be caught by you program. An example is Stack Overflow.

```
public int getNumber() throws IOException {
  BufferedReader in = new BufferedReader(new
InputStreamReader(System.in));
  System.out.print("Enter an integer: ");
  try { return Integer.parseInt(in.readLine()); }
  catch (Exception e) {
    // for extreme problems!
    System.out.println("I'm outta here!");
    return 0;
    }
  catch (NumberFormatException e) {
    // the typical problem.
    System.out.println("I couldn't interpret!");
    return getNumber();
    }
  finally {
    // if all else fails!!
    return 1;
    }
}
```

(b) (5 pts) This method should read a string from the terminal, interpret it as an integer and returns that integer as its value. If a non-integer is entered, the method asks the user to enter an integer again. Point out the errors in the method as written and correct it by crossing out and/or adding code.

As written, the first catch clause will be the one that is always used. So it should either be removed or at least put after the second catch clause. The finally clause should be removed, otherwise the function will always return 1, which is not what is intended.