# CS 1331 Exam 1 Practice

- Signing signifies you are aware of and in accordance with the **Academic Honor Code of Georgia Tech**.
- Calculators and cell phones are NOT allowed.
- This is an object-oriented programming test. Java is the required language. Java is case-sensitive. DO NOT WRITE IN ALL CAPS. A Java program in all caps will not compile. Good variable names and style are required. Comments are not required.

| Question | Points per Page | Points Lost | Points Earned | Graded By |
|---|---|---|---|---|
| Page 1 | 28 | - | = | |
| Page 2 | 10 | - | = | |
| Page 3 | 10 | - | = | |
| Page 4 | 10 | - | = | |
| Page 5 | 10 | - | = | |
| Page 6 | 10 | - | = | |
| Page 7 | 10 | - | = | |
| Page 8 | 10 | - | = | |
| Page 9 | 20 | - | = | |
| Page 10 | 20 | - | = | |
| Page 11 | 10 | - | = | |
| Page 12 | 20 | - | = | |
| TOTAL | 168 | - | = | |

1. **True or False**

In each of the blanks below, write "T" if the statement beside the blank is true, "F" otherwise.

[1] (a) ____ Java identifiers can contain letters, digits, and the underscore symbol and may start with a digit.

[1] (b) ____ The statement `int x = 3f/4f;` will compile, but the result will be truncated so that `x` gets the value `0`.

[1] (c) ____ In a for loop header, `for (`*initializer*`;` *condition*`;` *update*`)`, the Java compiler requires *initializer* to initialize a loop variable and *update* to update it.

[1] (d) ____ The declarations `double scores[]` and `double[] scores` are equivalent.

[1] (e) ____ Java arrays have a variable number of elements of mixed types.

[1] (f) ____ Given an array named `scores`, the statement `scores[scores.length + 1]` will not compile.

[1] (g) ____ Instance methods of a class can be called without first instaniating an object of the class.

[1] (h) ____ Every Java class has a default no-arg constructor in addition to the constructors you write yourself.

[2] (i) ____ In Java, every class you write is a subclass of at least one other class.

[2] (j) ____ In a constructor, if an explicit `super` call is present, it must be the first statement in the constructor.

[2] (k) ____ If a class defines a single constructor, the constructor contains an implicit `super` call if no explicit `super` call is provided in the constructor.

[2] (l) ____ You can define a subclass of an abstract class without defining any of the abstract methods defined in the superclass.

[2] (m) ____ In a concrete class that implements an interface, you must provide definitions for all of the methods declared in the interface.

[2] (n) ____ Overloading a superclass method in a subclass means defining a method with the same name as the superclass method but with a different parameter list.

[2] (o) ____ `protected` members are visible to classes in the same package and to subclasses.

[2] (p) ____ `private` members are visible in the class in which they are defined, but not in subclasses.

[2] (q) ____ `FileNotFoundException` is a checked exception.

[2] (r) ____ In a try statement with multiple `catch` clauses, the first `catch` clause that can catch the exception thrown in the corresponding `try` block will be exectued.

2. **Expression Evaluation**

For each expression below, write the value and then the Java data type of the evaluated legal expression in the space provided. Be exact. The type you give must be the **exact spelling of a Java primitive type including uppercase vs lowercase as it would appear in your program.**

Expression: `7 / 2`

[1] (a) Calculated value: _____

[1] (b) Java primitive type: _____

Expression: `64 - 16 * 2`

[1] (c) Calculated value: _____

[1] (d) Java primitive type: _____

Expression: `2.5f + 3.0 - 1.5f`

[1] (e) Calculated value: _____

[1] (f) Java primitive type: _____

Expression: `new Double(1) / 2`

[1] (g) Calculated value: _____

[1] (h) Java primitive type: _____

Expression: `true && "Foo".equals("foo")`

[1] (i) Calculated value: _____

[1] (j) Java primitive type: _____

3. **Multiple Choice** Circle the letter of the correct choice.

Given:

```
public class Kitten {

    private String name = "";

    public Kitten(String name) {
        name = name;
    }

    public String toString() {
        return "Kitten: " + name;
    }

    public boolean equals(Object other) {
        if (this == other) return true;
        if (null == other) return false;
        if (!(other instanceof Kitten)) return false;
        Kitten that = (Kitten) other;
        return this.name.equals(that.name);
    }
}
```

Assume the following statements have been executed:

```
Kitten maggie = new Kitten("Maggie");
Kitten fiona = new Kitten("Fiona");
Kitten fiona2 = new Kitten("Fiona");
Kitten mittens = fiona;
```

[2] (a) What is the value of `maggie`?

A. the address of a `Kitten` object
B. null
C. automatically set to 0
D. undefined

[2] (b) What is printed on the console after the following statement is executed?

```
System.out.println(maggie.toString());
```

A. Kitten:
B. Kitten: null
C. Kitten: Maggie

[2] (c) What is the value of the expression `fiona == mittens)`?

A. true
B. false

[2] (d) What is the value of the expression `fiona.equals(fiona2) && fiona == mittens`?

A. true
B. false

[2] (e) After executing `Kitten[] kittens = new Kitten[5];` , what is the value of `kittens[0]` ?

A. null
B. the address of a `Kitten` object
C. automatically set to 0
D. undefined

4. **Multiple Choice** Circle the letter of the correct choice.

[2] (a) In which package is `Object` from the standard library located?

A. java.util
B. java.lang
C. java.text
D. java.object

[2] (b) In a class named `Pill`, what is the correct declaration for a method that overrides the `equals` method defined in `Object`?

A. public boolean equals(Pill other)
B. public boolean equals(Object other)
C. protected boolean equals(Pill other)
D. protected static boolean equals(Object other)

[2] (c) A method declared in a superclass is said to be polymorphic in its subclasses if _____.

A. the method is declared `final` in the superclass
B. the method is overriden in the subclasses
C. the method is overloaded in the subclasses
D. the method chains to the superclasses using `super`

[2] (d) Which of the following features is required for a language to be called an object-oriented langauge?

A. separate compilation
B. dynamic method binding
C. lazy evaluation
D. higher-order functions

[2] (e) How many classes may a class extend?

A. 0
B. 1
C. 2
D. $[0, \infty)$

5. **Multiple Choice** Circle the letter of the correct choice.

Given the following class definitions:

```
public abstract class Animal {
  public abstract void speak();
}
```

```
public class Mammal extends Animal {
  public void speak() {
    System.out.println("Hello!");
  }
}
```

```
public class Dog extends Mammal {
  public void speak() {
    System.out.println("Woof, woof!");
  }
}
```

```
public class Cat extends Mammal {
  public void speak() {
    System.out.println("Meow!");
  }
}
```

[2] (a) Which of the following statements will **not** compile?

A. Animal mittens = new Cat();
B. Animal house = new Animal();
C. Animal farm = new Mammal();

[2] (b) Which of the following statements will **not** compile?

A. Mammal fido = new Dog();
B. Dog fido2 = fido;
C. ((Mammal) fido).speak();

[2] (c) Assuming the statement `Mammal fido = new Dog();` has been executed, what does `fido.speak()` print?

A. Hello!
B. Woof! Woof!
C. Meow!

[2] (d) Assuming the statement `Mammal fido = new Dog();` has been executed, what does `((Mammal) fido).speak()` print?

A. Hello!
B. Woof! Woof!
C. Meow!

[2] (e) Assuming the statement `Mammal sparky = new Mammal();` has been executed, which of the following statements will compile but cause a `ClassCastException` at run-time?

A. Mammal fido = new Dog();
B. Dog huh = (Dog) sparky;
C. Dog fido2 = (Dog) new Dog();

6. **Tracing**

Consider the following code:

```
public class StrangeLogic {
    private static int counter = 0;

    private static boolean incrementCounter() {
        counter++;
        return true;
    }
    public static void main(String args[]) {
        boolean a = true, b = false;
        if (b || incrementCounter()) {
            System.out.print("Boo");
        }
        if ((a || b) && incrementCounter()) {
            System.out.print(" ya!");
        }
        System.out.println(counter);
    }
}
```

[5] (a) What is printed when main is executed?

Consider the following code:

```
public class AlGore {

    public static void main(String[] args) {
        String mystery = "mnerigpaba";
        String solved = "";
        int len = mystery.length();
        for (int i = 0, j = len - 1; i < len/2; ++i, --j) {
            solved += mystery.charAt(i) + mystery.charAt(j);
        }
        System.out.println(solved);
    }
}
```

[5] (b) What is printed when main is executed?

[10] 7. **Tracing**

Consider the following code:

```
public class Wee {

    static void bar() throws Throwable {
        throw new Throwable("Wee!");
    }

    static void foo() throws Throwable {
        bar();
        System.out.println("Foo!");
    }

    public static void main(String[] args) {
        try {
            foo();
        } catch (Throwable t) {
            System.out.println(t.getMessage());
        }
        System.out.println("I'm still running.");
    }
}
```

What is printed when main is executed?

[10] 8. **Tracing**

Given the following class definitions:

```
public class Super {
  protected int x = 1;

  public Super() {
    System.out.print("Super");
  }
}
```

```
public class Duper extends Super {
  protected int y = 2;

  public Duper() {
    System.out.println(" duper");
  }
}
```

```
public class Fly extends Super {
  private int z, y;

  public Fly() {
    this(0);
  }

  public Fly(int n) {
    z = x + y + n;
    System.out.println(" fly times " + z);
  }

  public static void main(String[] args) {
    Duper d = new Duper();
    int delta = 1;
    Fly f = new Fly(delta);
  }
}
```

What is printed when `Fly` is run?

9. **Short Answer**

[2] (a) Assume you have a Java class named `Foo` that you want to be able to run from the command line. Write the header for the method you need to define in `Foo` to make it executable from the command line.

[2] (b) Assume you are at the command line in the directory of the file that contains the definition for a Java class named `Foo`. Write the command that you would execute on the command line to compile `Foo`.

[2] (c) If the command above executes successfully, what file will be produced?

[2] (d) Write the command that will execute the `Foo` class you compiled above.

[3] (e) What will the following code print?

```
for (int i = 10; i > 0; i--);
    System.out.print(‘‘Meow! ’’);
```

[4] (f) Assume you have the following start of a Student class and Student constructor. Finish the constructor. Do not change the code that is given.

```
public class Student {
    private String name;
    private String email;
    public Student(String name, String email) {
        // Your code goes here



    }
```

[5] (g) Convert the following `for` loop to an equivalent `while` loop.

```
for (int i = 10; i > 0; i--) {
    System.out.println(i);
}
```

10. **Short Answer**

[4] (a) Write the header for a class named `Foo` that extends a class called `Bar` and implements two interfaces, `Baz` and `Bang`.

[4] (b) Assume you have two variables of type `Foo` and `Foo` is properly written. The variables are named `f1` and `f2`. Write the expression that represents whether or not the objects that `f1` and `f2` reference have the same value, by the `Foo` class's definition of equal value.

[4] (c) Assume you have two variables of type `Foo` and `Foo` is properly written. The variables are named `f1` and `f2`. Write the expression that represents whether `f1` is an alias of `f2`.

[4] (d) Given that `FileInputStream`'s constructor throws `FileNotFoundException`, which is a subclass of `Exception`, write the header for a public method named `process` that takes a `String` parameter and returns nothing, and whose body instantiates a `FileInputStream` object and does not contain a try-catch statement.

[4] (e) Given a method declared as:

```
private void initFromFile(File empData) throws FileNotFoundException,
                                               IOException,
                                               ParseException
```

And the following declarations for the exception classes:

```
public class FileNotFoundException extends IOException
public class IOException extends Exception
public class ParseException extends Exception
```

Write a `try-catch` statement in which you call the `initFromFile` method and catch all the possible exceptions that might be thrown from `initFromFile`. Leave your catch clauses empty.

11. **Complete the Method**

[5] (a) Fill in the code for the following method that takes an array of numbers and returns the number of even numbers in the array argument. Your code should use a `for` loop.

```
public int evens(int[] numbers) {
    // Your code goes here









}
```

[5] (b) Fill in the code for the following method that takes an array of numbers and a number and returns `true` if the array contains the number, `false` otherwise. You will need a loop, and your loop must not execute more iterations than necessary, and you cannot use `break` or `continue`.

```
public boolean contains(int[] numbers, int n) {
    // Your code goes here









}
```

[20] 12. Given the following class and interface definitions:

```
public abstract class Pfunker implements Comparable {
    /**
     * LOLLYPOP < ATLANTEAN < CLONE < PILL < PYRAMID < FLASHLIGHT < ATOMIC_DOG
     */
    public enum Level {LOLLYPOP, ATLANTEAN, CLONE, PILL, PYRAMID,
                       FLASHLIGHT, ATOMIC_DOG}
    private Level level;
    private String name;

    public Pfunker(String name, Level level) {
        this.name = name;
        this.level = level;
    }
}
```

```
public interface Comparable {
    /**
     * Compares this object with the specified object for order. Returns a
     * negative integer, zero, or a positive integer as this object is less
     * than, equal to, or greater than the specified object.
     */
    public int compareTo(Object o);
}
```

Write the minimum concrete class named `ConcretePfunker` which is a subclass of `Pfunker`. You compare one `Pfunker` to another by comparing their `levels`. The space provided is more than sufficient. You will not be given any scratch paper. Hints:

- You may want to use `Enum`'s `ordinal()` method, which "Returns the ordinal [int] of this enumeration constant (its position in its enum declaration, where the initial constant is assigned an ordinal of zero)."
- The body of the one non-constructor method you need to write can be done in one line.