CS 1331 Exam 2 Practice

Name (print clearly): ________________________________________________

Signature: _________________________________________________________

GT account username (gtg, gth, msmith3, etc): _______________________

- 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.

<table>
<thead>
<tr>
<th>Question</th>
<th>Points per Page</th>
<th>Points Lost</th>
<th>Points Earned</th>
<th>Graded By</th>
</tr>
</thead>
<tbody>
<tr>
<td>Page 1</td>
<td>20</td>
<td>-</td>
<td>=</td>
<td></td>
</tr>
<tr>
<td>Page 2</td>
<td>10</td>
<td>-</td>
<td>=</td>
<td></td>
</tr>
<tr>
<td>Page 3</td>
<td>10</td>
<td>-</td>
<td>=</td>
<td></td>
</tr>
<tr>
<td>Page 4</td>
<td>10</td>
<td>-</td>
<td>=</td>
<td></td>
</tr>
<tr>
<td>Page 5</td>
<td>10</td>
<td>-</td>
<td>=</td>
<td></td>
</tr>
<tr>
<td>Page 6</td>
<td>20</td>
<td>-</td>
<td>=</td>
<td></td>
</tr>
<tr>
<td>Page 7</td>
<td>20</td>
<td>-</td>
<td>=</td>
<td></td>
</tr>
<tr>
<td>TOTAL</td>
<td>100</td>
<td>-</td>
<td>=</td>
<td></td>
</tr>
</tbody>
</table>
1. **True or False**

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

(a) ___ In Java, every class you write is a subclass of at least one other class.

(b) ___ In a constructor, if an explicit `super` call is present, it must be the first statement in the constructor.

(c) ___ If a class defines a single constructor, the constructor contains an implicit `super` call if no explicit `super` call is provided in the constructor.

(d) ___ You can define a subclass of an abstract class without defining any of the abstract methods defined in the superclass.

(e) ___ In a concrete class that implements an interface, you must provide definitions for all of the methods declared in the interface.

(f) ___ 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.

(g) ___ *protected* members are visible to classes in the same package and to subclasses.

(h) ___ *private* members are visible in the class in which they are defined, but not in subclasses.

(i) ___ `FileNotFoundException` is a checked exception.

(j) ___ 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 executed.
2. **Multiple Choice** Circle the letter of the correct choice.

(a) In which package is `Object` from the standard library located?
   - A. `java.util`
   - B. `java.lang`
   - C. `java.text`
   - D. `java.object`

(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)`

(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 overridden in the subclasses
   - C. the method is overloaded in the subclasses
   - D. the method chains to the superclasses using `super`

(d) Which of the following features is required for a language to be called an object-oriented language?
   - A. separate compilation
   - B. dynamic method binding
   - C. lazy evaluation
   - D. higher-order functions

(e) How many classes may a class extend?
   - A. 0
   - B. 1
   - C. 2
   - D. `[0, \infty)`
3. Multiple Choice Circle the letter of the correct choice.

Given the following class definitions:

```java
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!");
    }
}
```

(a) Which of the following statements will **not** compile?
A. Animal mittens = new Cat();
B. Animal house = new Animal();
C. Animal farm = new Mammal();

(b) Which of the following statements will **not** compile?
A. Mammal fido = new Dog();
B. Dog fido2 = fido;
C. ((Mammal) fido).speak();

(c) Assuming the statement `Mammal fido = new Dog();` has been executed, what does `fido.speak()` print?
A. Hello!
B. Woof! Woof!
C. Meow!

(d) Assuming the statement `Mammal fido = new Dog();` has been executed, what does `((Mammal) fido).speak()` print?
A. Hello!
B. Woof! Woof!
C. Meow!

(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();
4. Tracing

Consider the following code:

```java
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?
5. **Tracing**

Given the following class definitions:

```java
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?
6. **Short Answer**

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

(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.

(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`.

(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.

(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.
7. Given the following class and interface definitions:

```java
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.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-constructors method you need to write can be done in one line.