COMP 401
Spring 2014
Midterm 2

I have not received nor given any unauthorized assistance in completing this exam.

Signature: ____________________________

Name: _______________________________

PID: ________________________________

Please be sure to put your PID at the top of each page.
PID: ______________________

This page left intentionally blank. Do not write on it.
Part I: Multiple choice (40 points total)

Directions:
For each multiple choice question below, indicate all correct answers among the
choices provided. More than one answer may be correct and at least one answer
will be correct. Each question is worth 5 points. Selecting all of the choices or
selecting none of the choices will result in no points awarded.

1. Which of these statements about subclass access to parent class fields is true?
a) A subclass always has direct access to private fields defined in a parent class.
b) A subclass always has direct access to protected fields defined in a parent class.
c) A subclass always has direct access to public fields defined in a parent class.
d) A subclass always has direct access to unmarked fields (i.e., those with no access
   modifier) defined in a parent class.
e) A subclass must redeclare the parent class fields that it needs to access.

2. Which of these statements about inheritance is true?
a) Inheritance can be used to eliminate duplicate code in related subclasses.
b) A subclass can inherit from more than one parent class.
c) A subinterface can inherit from more than one parent interface.
d) A subclass does not automatically have an is-a relationship with any interfaces
   implemented by its parent class.
e) A subclass must define at least one new method that is not defined by its parent
   class.

3. Which of these statements about overriding is true?
a) All overridden methods in Java are virtual.
b) An overriding method defined in a subclass must have a method signature that is
   somehow different from the method defined in the parent class.
c) An overridden method in a subclass is able to call the same method defined in the
   parent class by using the this keyword.
d) Overriding is related to the principle of encapsulation.
e) It is not possible to override a private method.

4. Which of these statements about Composition and Aggregation are true?
a) Providing encapsulated objects as parameters to a constructor is a characteristic
   of aggregation.
b) Providing getters and setters for its encapsulated objects is a characteristic of
   composition.
c) An object that encapsulates other objects must either be a composition or an
   aggregation.
d) An aggregation or a composition can claim an is-a relationship with an interface
   implemented by one of its encapsulated objects by using delegation.
e) Composition and aggregation are design techniques that rely on inheritance.
5. Which of these statements about exceptions are true?
   a) An exception that is a subclass of RuntimeException is subject to the “catch-or-specify” policy.
   b) The finally block of a try-catch-finally structure is executed only when an exception occurs.
   c) A catch block associated with the type Exception should never occur before a catch block associated with the type RuntimeException.
   d) An exception must be caught as soon as possible.
   e) A try-catch-finally structure can be nested within a catch block.

6. Which of these statements about the Factory design pattern are true?
   a) A class employing the Factory design pattern will usually have at least one public constructor.
   b) A class employing the Factory design pattern will usually have at least one public static method.
   c) The Factory design pattern is useful for creating singletons.
   d) The Factory design pattern can be useful for dynamically choosing among several subclasses when creating a new instance of a common parent class.
   e) The Factory design pattern is an example of delegation.

7. Which of these statements about Observer/Observable are true?
   a) An observable class usually provides a public method for registering observers.
   b) All observer classes of a particular observable class will share a common parent class.
   c) An observable class should be immutable.
   d) All observer classes of a particular observable class will implement a common interface.
   e) Observers are always updated in the same order as they were registered.

8. Which of these statements about the Decorator design pattern are true?
   a) The Decorator design pattern uses subclass inheritance to add additional state or functionality to an existing class.
   b) The Decorator design pattern relies on delegation.
   c) The Decorator design pattern requires the original class to be associated with an interface.
   d) The constructor of a class using the Decorator design pattern should not require any parameters.
   e) A class using the Decorator design pattern will usually need to declare one or more private instance fields to encapsulate object state.
Part II: Is-A Relationships (10 points total)

Suppose the following interfaces and classes are declared (the actual definitions are unimportant)

```java
public interface I1
public interface I2 extends I1
public interface I3
public interface I4 extends I3

public class A implements I1
public class B extends A implements I2
public class C extends A implements I4
public class D extends B implements I3
public class E implements I2
public class F extends E implements I3
public class G extends E implements I4
```

Suppose the following variables are defined:

```java
A obj_a = new A();
B obj_b = new B();
C obj_c = new C();
D obj_d = new D();
E obj_e = new E();
F obj_f = new F();
G obj_g = new G();
```

For each of the following statements, draw a line through any of the statements that are not legal (i.e., would either not compile or cause a runtime exception).

```java
I1 ref_i1 = (I1) obj_g;
I2 ref_i2 = (I2) obj_d;
I3 ref_i3 = (I3) obj_d;
I4 ref_i4 = (I4) obj_a;
A ref_a = (A) obj_d;
B ref_b = (B) obj_c;
C ref_c = (C) obj_g;
D ref_d = (D) obj_f;
E ref_e = (E) obj_g;
G ref_g = (G) obj_e;
```
Part III: Evaluating Code (2 points for each part a-e, 10 points total)

Given the following class definitions:

```java
class Foo {
    public int m (int a) {
        return m(a+1, a-1);
    }

    public int m (double a) {
        return m ((int) (a+1.0));
    }

    public int m (int a, int b) {
        return a-b;
    }
}

class Bar extends Foo {
    public int m (int a) {
        return super.m(a+3);
    }

    public int m (int a, int b) {
        return a+b;
    }
}

class Doh extends Bar {
    public int m (double a) {
        return m((int) (a+2.0));
    }

    public int m (int a) {
        return m(a-1, a+2);
    }
}
```
Suppose the following variables are defined:

```java
Bar b = new Bar();
Doh d = new Doh();
```

What is the value of the following expressions:

a) d.m(1, 2)

b) b.m(2)

c) d.m(2)

d) b.m(2.0)

e) d.m(2.0)
Part IV: Exceptions (2 points for each part a-e, 10 points total)

Given the following code defining four exception classes and two functions:

```java
class ExceptionA extends RuntimeException { }
class ExceptionB extends ExceptionA { }
class ExceptionC extends RuntimeException { }
class ExceptionD extends ExceptionB { }

class ExceptionA extends RuntimeException { }

class ExceptionB extends ExceptionA { }

class ExceptionC extends RuntimeException { }

class ExceptionD extends ExceptionB { }

public int bar(int x) {
    switch (x) {
        case 1:
            throw new ExceptionA();
        case 2:
            throw new ExceptionB();
        case 3:
            throw new ExceptionC();
        case 4:
            throw new ExceptionD();
        default:
            return x+2;
    }
}

public int foo(int y) {
    int x = 0;
    try {
        x += bar(y);
        x += 1;
    } catch (ExceptionA e1) {
        try {
            x += bar(y*2);
        } catch (ExceptionB e2) {
            x += 2;
        }
        x += 2;
    } catch (ExceptionC e1) {
        x += 3;
    } catch (RuntimeException e1) {
        x += 4;
    } finally {
        x += 1;
    }
    return x;
}
```
Part IV continued...

What is the value of the following expressions:

a) $\text{foo}(1)$

b) $\text{foo}(2)$

c) $\text{foo}(3)$

d) $\text{foo}(4)$

e) $\text{foo}(5)$
Part V: Inheritance (20 points)

Refactor the following classes Truck, Car, and Motorcycle employing inheritance as appropriate given the following guidelines:

- Car and Motorcycle should be subclasses of a common parent class called NonCommercial.
- NonCommercial and Truck should be subclasses of a common parent class called Vehicle.

Write your code on pages 13 and 14.

class Truck {
    private String license_plate;
    private String operator;
    private int number_of_axles;

    public Truck(String license_plate, String operator, int number_of_axles) {
        this.license_plate = license_plate;
        this.operator = operator;
        this.number_of_axles = number_of_axles;
    }

    public String getPlate() {
        return license_plate;
    }

    public String getOperator() {
        return operator;
    }

    public int getNumberOfAxles() {
        return number_of_axles;
    }

    public String getHornSound() {
        return "HONK";
    }
}

class Car {
    private String license_plate;
    private String make;
    private String model;

    public Car (String license_plate,
                String make,
                String model) {
        this.license_plate = license_plate;
        this.make = make;
        this.model = model;
    }

    public String getPlate() {
        return license_plate;
    }

    public int getNumberOfAxles() {
        return 2;
    }

    public String getMake() {
        return make;
    }

    public String getModel() {
        return model;
    }

    public String getHornSound() {
        return "Honk";
    }
}
class Motorcycle {
    private String license_plate;
    private String make;
    private String model;

    public Motorcycle(String license_plate,
                       String make,
                       String model) {
        this.license_plate = license_plate;
        this.make = make;
        this.model = model;
    }

    public String getPlate() {
        return license_plate;
    }

    public int getNumberOfAxles() {
        return 2;
    }

    public String getMake() {
        return make;
    }

    public String getModel() {
        return model;
    }

    public String getHornSound() {
        return "beep";
    }
}