COMP 401
Spring 2013
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.
This page left intentionally blank. Do not write on it.
Part I: True/False (2 points per question, 40 points total)

T____ A class that implements a subinterface must also implement the parent interface.

F____ A subinterface can extend more than one parent interface.

F____ If a class implements an interface, you must define subinterfaces for any subclasses of the class.

F____ A parent class has direct access to protected fields within a subclass.

T____ Defining a method in a subclass with the same signature as a method in the parent class is known as overriding.

F____ Overriding is an example of polymorphism.

F____ A subclass inherits only private instance variables from the parent class.

T____ A method defined by a subclass is able to invoke a protected method defined in the parent class.

T____ A call to super can be invoked only as the first statement of a subclass constructor.

T____ If a class contains an abstract method, it must be an abstract class.

F____ Casting a reference to an object from a subclass type to a parent class type is an example of contravariance.
T___ Composition and Aggregation are design techniques for defining objects as an encapsulation of other objects.

F___ Component objects within an aggregation are not typically useful outside of the aggregation.

T___ Encapsulated objects of an aggregation are usually provided as parameters to the constructor.

T___ Composition objects generally do not provide getters and setters for encapsulated objects.

F___ The Factory design pattern employs delegation.

F___ The Factory design pattern is useful when a class needs to provide a number of polymorphic public constructors.

T___ Dynamic subclass binding is an example of the Factory design pattern.

F___ An exception that is a subclass of RuntimeException is subject to the catch or specify requirement.

F___ An abstract class can be instantiated directly.
Part II: Is-A Relationships (10 points total)

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

```java
public interface InterA
public interface InterB extends InterA
public interface InterC extends InterA
public interface InterD extends InterC
public interface InterE
public interface InterF extends InterE

public class A implements InterA
public class B implements InterB
public class C extends A implements InterF
public class D extends B implements InterD
public class E extends A implements InterE
public class F implements InterF
```

Suppose the following variables are defined:

```java
A a1 = new A();
B b1 = new B();
C c1 = new C();
D d1 = new D();
E e1 = new E();
F f1 = new F();
```

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
InterA ia1 = (InterA) d1;
InterB ib1 = (InterB) e1;
InterC ic1 = (InterC) c1;
InterD id1 = (InterD) b1;
InterE ie1 = (InterE) c1;
InterF if1 = (InterF) f1;
A a2 = (A) b1;
B b2 = (B) d1;
C c2 = (C) f1;
F f2 = (F) e1;
```
Part III: Evaluating Code (2 points for each part a-e, 10 points total)

Given the following class definitions:

```java
public class Foo {
    public int m(int x) {
        return x*2;
    }
}

public class Bar extends Foo {
    public double m(double x) {
        return x*4.0;
    }
}

public class Doh extends Bar {
    public int m(int y) {
        return y + super.m(y);
    }
}
```

Suppose the following variables are defined:

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

What is the value of the following expressions:

a) f.m(5)  
   10

b) b.m(5)  
   10

c) b.m(1.5)  
   6.0

d) d.m(2.0)  
   8.0

e) d.m (2)  
   6
Part IV: Exceptions (2 points for each part a-d, 4 points for part e, 10 points total)

Given the following code:

class ExceptionA extends RuntimeException { }
class ExceptionB extends ExceptionA { }
class ExceptionC extends RuntimeException { }

class ExceptionA extends RuntimeException { }
class ExceptionB extends ExceptionA { }
class ExceptionC extends RuntimeException { }

public int bar(int x) {
    if (x == 1) {
        throw new ExceptionA();
    }
    if (x == 2) {
        throw new ExceptionB();
    }
    if (x == 3) {
        throw new ExceptionC();
    }
    return 2*x;
}

public int foo(int y) {
    int x = 0;
    try {
        x += bar(y);
        x += 4;
    } catch (ExceptionB e) {
        x += 5;
    } catch (ExceptionA e) {
        x += 6;
    } catch (RuntimeException e) {
        x += 7;
    } finally {
        x += 8;
    }
    return x;
}

What is the value of the following expressions:
a) foo(1) 14
b) foo(2) 13
c) foo(3) 15
d) foo(4) 20
e) What will the following program print?

class ExceptionA extends Exception { }
class ExceptionB extends ExceptionA { }

public class Test {
    public static void main (String[] args) {
        boolean exit = false;
        try {
            throw new ExceptionB();
        } catch (ExceptionA e) {
            System.out.println("Go Heels!");
            exit = !exit;
        } catch (Exception e) {
            System.out.println("Beat Duke!");
            exit = !exit;
        } finally {
            if (exit) {
                return;
            }
        }
        System.out.println("COMP 401 rocks!");
    }
}

Go Heels!
Part V: Inheritance (20 points)

Refactor the following classes Undergrad, GradStudent, and Professor employing inheritance as much as possible by making them all subclasses from a common parent class called Person. Write your code on page 12.

```java
public class Undergrad {
    private String name;
    private int PID;
    private String major;

    public Undergrad(String name, int PID, String major) {
        this.name = name;
        this.PID = PID;
        this.major = major;
    }

    public String getName() {
        return name;
    }

    public int getPID() {
        return PID;
    }

    public String getMajor() {
        return major;
    }

    public String getGreeting() {
        return "What's up?";
    }
}
```
public class GradStudent {
    private String name;
    private int PID;
    private boolean phd_intending;

    public GradStudent(String name, int PID, boolean phd_intending) {
        this.name = name;
        this.PID = PID;
        this.phd_intending = phd_intending;
    }

    public String getName() {
        return name;
    }

    public int getPID() {
        return PID;
    }

    public boolean isPhDIntending() {
        return phd_intending;
    }

    public String getGreeting() {
        return "Hello";
    }
}
public class Professor {
    private String name;
    private int PID;
    private String rank;

    public GradStudent(String name, int PID, String rank) {
        this.name = name;
        this.PID = PID;
        this.rank = rank;
    }

    public String getName() {
        return name;
    }

    public int getPID() {
        return PID;
    }

    public String getRank() {
        return rank;
    }

    public String getGreeting() {
        return "Live long and prosper";
    }
}
public abstract class Person {
    private String name;
    private int PID;

    public Person(String name, int PID) {
        this.name = name;
        this.PID = PID;
    }

    public String getName() {
        return name;
    }

    public int PID() {
        return PID;
    }

    abstract public String getGreeting();
}

public class Undergrad {
    private String major;

    public Undergrad(String name, int PID, String major) {
        super(name, PID);
        this.major = major;
    }

    public String getMajor() {
        return major;
    }

    public String getGreeting() {
        return "What's up?";
    }
}

public class GradStudent {
    private boolean phd_intending;

    public GradStudent(String name, int PID, boolean phd_intending) {
        super(name, PID);
        this.phd_intending = phd_intending;
    }

    public boolean isPhDIntending() {
        return phd_intending;
    }
public class Professor {
    private String rank;

    public Professor(String name, int PID, String rank) {
        super(name, PID);
        this.rank = rank;
    }

    public String getRank() {
        return rank;
    }

    public String getGreeting() {
        return "Live long and prosper";
    }
}