Part 1. Syntax of Inheritance

1. Declare that Student is a subclass of Person. Write your answer in the code.

2. Complete the Student constructor. Write your answer in the code.

public class **Person** {

private String **name**;

public **Person(String name)** {

this.name = name;

}

public String **getName**() { return name; }

public void **setName(String newname)** { this.name = newname; }

public String **toString**() { return "Person named "+name; }

}

public class **Student**

{

private Long studentId;

/\*\*

\* Constructor for a student with a name and id number.

\*/

public **Student**(String name, Long id) {

}

public String toString() {

return String.format("Student %s (%d)", getName(), studentId);

}

}

3. What is printed by this code:

Person p = new Student("Hacker", 5412345678L);

System.out.println( p.toString() );

a) "Person named Hacker"

b) "Student Hacker (5412345678)"

c) (a) is printed first, then (b)

d) Syntax error: can't assign Student to Person reference (p).

4. We want Student to *inherit* the getName() method of Person (Student is a subclass of Person). What should we write in the Student class? Circle the correct answer.

a) Don't write anything.

b) @Override

public String getName() { super(); }

c) @Inherit

public String getName() { return super.getValue(); }

d) @Override

public double getValue();

5. In the Person class, suppose we want to *ensure* that no subclass overrides the getName() method. What should we write in Person?

a) Don't write anything.

b) @Final

public String getName() { return this.name; }

c) public abstract String getName() { return this.name; }

d) public final String getName() { return this.name; }

e) None of these. A subclass can always override a superclass method.

6. Suppose Coin is a subclass of Money, and Money a subclass of Object. We write

**Object**

# clone( )

equals(Object)

getClass()

hashCode()

toString()

// *ignore other methods*

Coin coin = new Coin( 10 );

(a) Using code inside the Coin class, which methods can we call?

Put a check mark ( ) next to all the methods from every class that coin can call using this or super, and X next to the methods that Coin cannot call.

(b) The Money class implements *Comparable* with parameter type Money. Show this relationship on the UML diagram.

(c) the value attribute of Money is double but the value of a Coin is an int.

**Money**

Money(value: double)

equals(Object)

getValue( ): double

toString()

Can Coin define its own getValue( ) method that returns int? Wny?

(d) Suppose that money.getValue( ) returns double and coin.getValue() returns int. Give example code to show how this might violate the *Liskov Substitution Principle*.

**Coin**

Coin( value: int )

equals(Object)

Part 2: Design with Inheritance

These exercises are described in "Inhteritance-Practice" document (not PDF) on class web.